TFM – DISSENY I IMPLEMENTACIÓ D'UNA ARQUITECTURA I SISTEMA DISTRIBUÏT MÒBIL/WEB: CAS DE E-COMMERCE PER IMATGES.

Versió 1

15/04/2019

Marc Tula Guardiola

Universitat Politècnica de Catalunya

Abstract

Aquest Treball Final de Màster té com a objectiu el realitzar una comparativa entre les diverses tecnologies actuals que existeixen per a la realització d'un projecte distribuït amb aplicació web i aplicació mòbil connectades a un Web Service així com detallar-les explicant els punts forts i febles, en quin entorn usar-les etc.

Realitzada la comparativa, l'objectiu és proposar un cas pràctic real que consisteix en la creació de Shoreshots, una plataforma de venta d'imatges de surfistes o practicants d'esports similars. El sistema complirà els requisits descrits i estarà composat per una aplicació web i una aplicació mòbil facilitant així l'accés als usuaris per a que puguin accedir al sistema des d'un dispositiu mòbil o bé, un ordinador. Les dues plataformes fan ús d'un Web Service on es troben definides i implementades totes les funcionalitats per a inserir i consultar la informació emmagatzemada. No és objectiu principal mostrar el sistema finalitzat sinó assentar les bases que descriguin els punts forts de la tecnologia escollida.

Els usuaris interessats podran pujar les seves imatges per a que els surfistes es cerquin a la plataforma i adquireixin la imatge per un preu. El projecte neix per una necessitat real dels amants d'aquest tipus d'esports que veuen com a altres països existeixen plataformes similars però aquí a Espanya encara no.

En aquest document s'explica tot el procés des de la idea original, la comparativa amb altres serveis similars, la recerca de tecnologies i metodologies i la implementació detallant les eines usades. Per últim s'analitzen els resultats obtinguts així com les línies de futur que pot tenir aquest projecte.

Paraules clau: Aplicació, Pàgina Web, Web Service, Surf, e-commerce

Abstract

Este Trabajo Final de Máster tiene como objetivo el realizar una comparativa entre les distintas tecnologías actuales que existen para la realización de un proyecto distribuido con aplicación web y aplicación móvil conectadas a un Web Service así como detallarlas explicando los puntos fuertes y débiles, en que entorno usarlas, etc.

Realizada la comparativa, el objetivo es proponer un caso práctico real que consiste en la creación de Shoreshots, una plataforma de venta de imágenes de surfistas o practicantes de deportes similares. El sistema cumplirá los requisitos descritos y estará compuesto por una aplicación web y una aplicación móvil facilitando así el acceso a los usuarios para que puedan acceder al sistema des de un dispositivo móvil o bien, un ordenador. Las dos plataformas hacen uso de un Web Service donde se encuentran definidas e implementadas todas las funcionalidades para insertar y consultar la información almacenada. No es objetivo principal mostrar el sistema finalizado sino asentar les bases que describan los puntos fuertes de la tecnología escogida.

Los usuarios interesados podrán subir sus imágenes para que los surfistas puedan buscarse en la plataforma y adquirir la imagen por un precio. El proyecto nace por una necesidad real de los amantes de este tipo de deportes que ven como en otros países existen plataformas similares pero aquí en España aún no.

En este documento se explica todo el proceso des de la idea original, la comparativa con otros servicios similares, la investigación de tecnologías y metodologías y la implementación detallando las herramientas usadas. Per último se analizan los resultados obtenidos así como las líneas de futuro que puede tener este proyecto.

Palabras clave: Aplicación, Página Web, Web Service, Surf, e-commerce

Abstract

This Final Master Work has the goal of making a comparison between the different current technologies available to develop a distributed system composed of a web application, mobile application and Web Service and also explain in detail the different methods, environments, strengths and weak spots, etc.

Once the comparison is made, the next goal is to propose a practical case which consists in the creation of Shoreshots, a surfer image online sale platform. The System will satisfy the specified requirements and will be composed of a web application and a mobile phone helping users to access the system from different devices. Both platforms consume a Web Service where all data is accessed and stored. It is not a main goal to present a finished project but to build the basics making emphasis in the strengths of the chosen technology.

The interested users will be able to upload their images so the surfers can search themselves and acquire the image for a price. The project is born from the real necessity of the lovers of this kind of sports that find other similar platforms in different countries but no yet in Spain or Catalonia.

In this document is explained all the process from the original idea trough the research and investigation to the implementation explaining in detail the methods and used tools. By last, the results and the future options of this project are analyzed.

Key words: Application, web application, Web Service, Surf, e-commerce

Agraïments

M'agradaria donar les gràcies al Victor Ferrer, Berni Nadal i Sandro Del Val per a donar-me l'oportunitat de participar en aquest projecte el qual m'ha permès ampliar els meus coneixements en desenvolupament web, Android i bases de dades. Treballar en un projecte real mostra tots els avantatges i inconvenients del món laboral i és una gran experiència que m'emporto.

També m'agradaria donar les gràcies al meu tutor, Jaime Delgado qui ha portat el seguiment del projecte i m'ha ajudat en tots els dubtes que han aparegut. Per últim donar les gràcies a la meva família i amics els quals m'han donat suport en tot el camí.

Índex

1.Introducció

1.1 Marc del treball

Avui en dia és impossible imaginar un món sense internet i sense mòbils. Cada vegada és més gran l´ús de plataformes web o aplicacions per a donar un servei, proporcionar entreteniment, informació o promocionar un negoci.

Segons l'informe de Pickaso: [1] Hábitos de Consumo Mobile en España y en el Mundo en 2018 – Emma Olivero, l'augment d'smartphones connectats a la xarxa al món augmenta cada any tal i com es veu a la Figura 1 i es preveu que segueixi creixent a un ritme més moderat. A l'any 2018 el nombre d'usuaris amb mòbil arriba fins a 5135 milions de persones o el que és el mateix, un 68% de la població mundial. Per altre banda 4021 milions de persones ja conten amb connexió a internet (penetració del 53%)

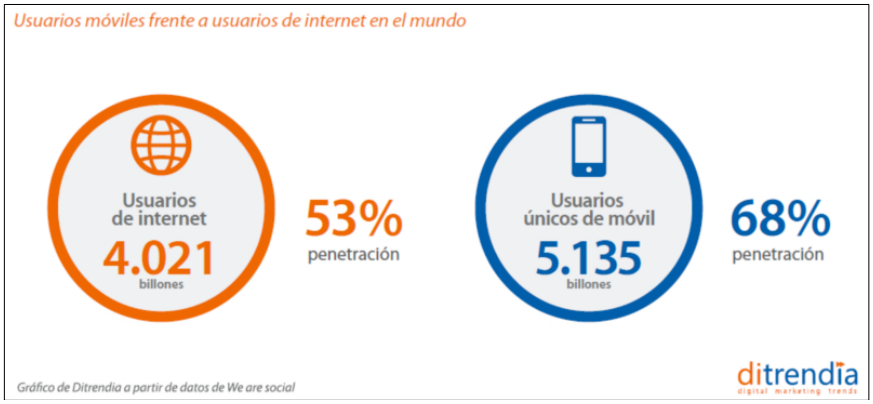


Figura 1 –Ús del Smartphone i Internet al 2018 [1]

L'ús d'aplicacions suposa un 54% del temps gastat al món digital i es dediquen de mitja 1.4 hores a tabletes i 3 des d'smartphones. És per això que a la hora de dur a terme un projecte de negoci o cultura és imprescindible el fet d' expandir- se a internet a pàgines web però també al món de les aplicacions per les diferents plataformes disponibles.

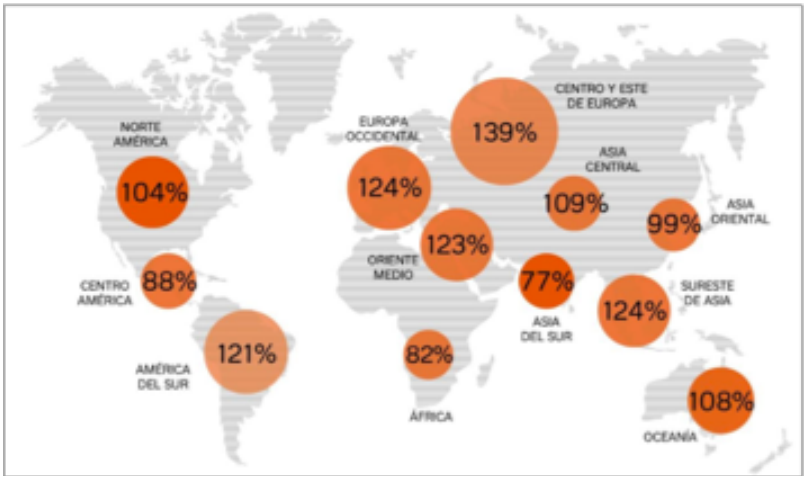


Figura 2 – Penetració del Mòbil al món [1]

A la Figura 2 s'observa la penetració del mòbil a diferents regions del món on només el sud d'Àsia, Àfrica i el centre americà té una penetració inferior al 100%.

L'ús del Smartphone per accedir a pàgines web augmenta any rere any, de fet a mercats com a Espanya ja hi han més usuaris mòbil que d'escriptori. A la Figura 3 s'observa el repartiment del trànsit web per dispositiu.

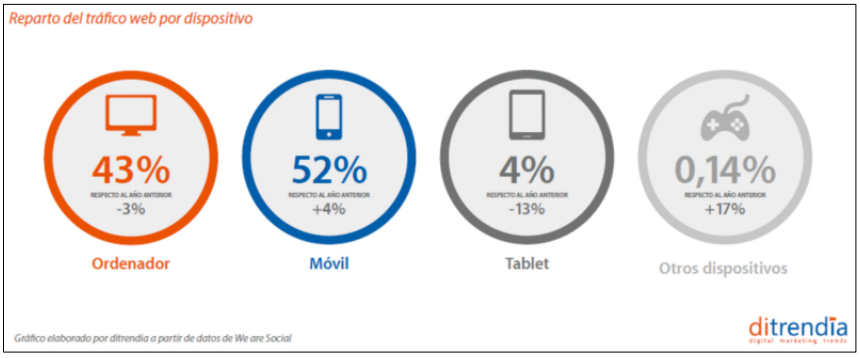


Figura 3 – Repartiment del trànsit web per dispositiu

Sembla clar doncs, que a l'hora de desenvolupar un projecte d'aplicació web aquesta haurà d'estar clarament adaptada a les diferents mides de pantalla *responsive-design* així com, segons el projecte, disposar d'una aplicació per a facilitar l'accés des d'aquests dispositius.

1.2 – El món del surf a Catalunya i Espanya

El treball descrit en aquests document consisteix en una comparativa de les maneres més òptimes per al disseny e implementació d'un sistema distribuït composat per una aplicació web i una aplicació mòbil connectades a un Web Service, així com la descripció de l'aplicació d'una d'aquestes tecnologies, la considerada més òptima, a un projecte real del món del surf.

En concret, és vol desenvolupar una eina online amb el nom de Shoreshots que permeti a fotògrafs pujar fotografies de surfistes a la plataforma per a que aquests es puguin buscar i adquirir la imatge per un preu.

De la mateixa manera que és un fet l'augment de l'ús d'Smartphones i consum web a Espanya i arreu del món, també ho és la pràctica d'activitats esportives relatives al Surf o els seus derivats: Kite-surf, Wind-surf, etc. Segons la revista GQ i l'article publicat per Marta Sahelices – "¿Por qué España se ha convertido en un país con 40 millones de surferos?"– a Espanya hi ha actualment més de 30.000 persones federades en aquests esports i molts milers més que la practiquen.

La tendència és un augment global en la pràctica d'aquest esport amb cada cop més influencia a països com Espanya i més ara, amb l'anunci del comitè olímpic internacional de que el Surf serà inclòs als jocs de Tokyo 2020.

Les imatges practicant aquest esport són difícils d'aconseguir i són molt buscades degut a que al practicar-se amb onades al mar acostumen a ser espectaculars. És per això que és molt habitual l'aparició de fotògrafs a les platges per a captar als surfistes practicant l'esport.

Projectes similars al que es vol dur a terme en aquest treball estan tenint força èxit a Portugal amb el portal *Surfterra*. Al capítol de l'estat s'estudia aquesta plataforma analitzant els punts forts i febles per a tenir amb compte de cara al desenvolupament del projecte.

1.3 – Estructura del document

En aquest apartat s'explica la estructura d'aquest Treball Final de Màster indicant cada un dels capítols que el composen amb una breu descripció de cada un.

- **2 - Estat de l'Art**: en aquest capítol s'estudia el mercat per a veure quins projectes similars existeixen i extreure els punts forts i febles de cada un.
- **3 - Especificació de Requisits Software**: en aquest capítol s'explica el projecte Surf And Share un cop estudiats els projectes similars. A més, s'incorpora el document d'especificació de requisits de software on queda definit tot l'abast del projecte.
- **4 - Fonaments Teòrics**: en aquest capítol es dur a terme una comparativa entre les diferents maneres de plantejar i desenvolupar un projecte d'aquestes característiques i finalment es justifica la tria d'una de les tecnologies i es documenten les eines necessàries per al desenvolupament.
- **5 - Desenvolupament**: en aquest capítol s'explica tot el procés de desenvolupament de cada un dels elements (pàgina web, aplicació, web service i base de dades) tot aportant exemples. S'explica des del disseny de les etapes fins a la implementació de codi.
- **6 – Costos del Projecte**: en aquest capítol s'aporta una gràfica temporal de les hores invertides en cada etapa així com una descripció de les tasques més rellevants realitzades a cada una.
- **7 – Conclusions**: en aquest capítol s'enumeren les diferents conclusions extretes durant la implementació del Treball Final de Màster.
- **8 – Línies de Futur**: en aquest capítol es proposen diverses opcions per a complementar aquest projecte i millorar-lo.
- **9 – Referències**: en aquest capítol s'enumeren les referencies que apareixen durant el document mostrant l'autor, l'any de la publicació i l'enllaç.

2- Estat de l'Art – Plataformes similars

En aquest apartat es mostren alguns projectes similars al que es presenta en aquest document. L'objectiu és el de veure com es troba el mercat dins aquest sector analitzant quins projectes estan funcionat i quins no.

Es tindran en compte els punts forts de les aplicacions que estiguin funcionant i s'evitarà caure en els punts febles. D'aquesta manera es tindrà una referència de per on pot anar el projecte i treure-li el màxim rendiment.

Per altre banda, també es tracta d'innovar en alguns aspectes de cara a diferenciar el projecte aquí descrit de la competència i que la pàgina i aplicació siguin les més utilitzades.

Surfterra

Surfterra [3] és un projecte molt similar al descrit en aquest document però per al territori portuguès. Es tracta d'un portal e-commerce on els surfistes poden buscar-se mitjançant uns filtres (Localització, data, etc.) i adquirir imatges per un preu.

Analitzant la pàgina s'observa un disseny atractiu amb un molt bon contrast de colors i adaptable per a totes les mides de pantalla (*Responsive-design*, veure Figura 9*)*. No hi ha una gran quantitat d'informació sinó que el producte principal, les imatges a la venta, son fàcilment accessibles per als usuaris.(veure Figures 6 i 7).

Es poden realitzar cerques específiques usant la barra de cerca del *banner* principal que funciona per paraula clau (Lloc, Fotògraf o Surfista registrat). En aplicar una cerca usant la barra del banner es troba un llistat d'imatges en cas d'haver entrat una localització reconeguda o bé un perfil de surfista o fotògraf. El perfil sembla públic i es troba informació com el nom, nacionalitat, visites, likes i followers. Existeix la opció de seguir (Follow) el perfil i apareixen les sessions, imatges i vídeos del fotògraf. (Veure Figura 8)

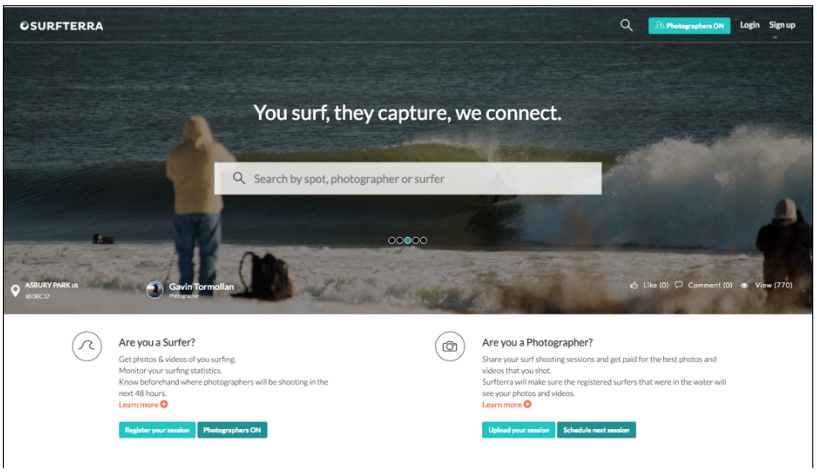


Figura 6 – Pàgina principal de la guia Woffi [3]

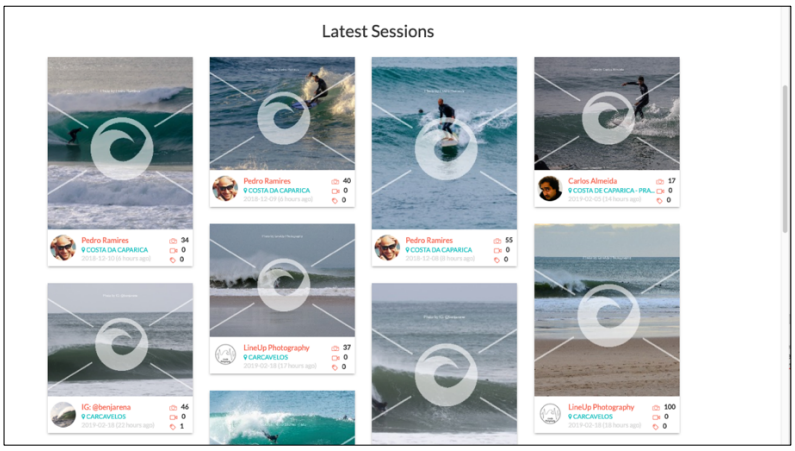


Figura 7 - Venta d'imatges a la página principal

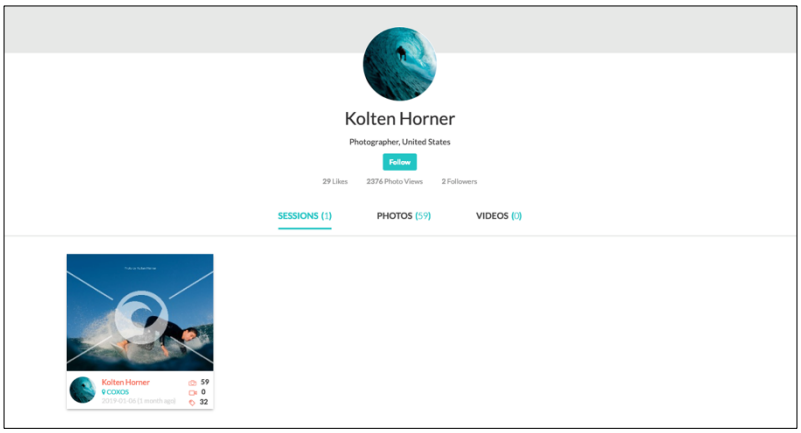


Figura 8 – Perfil fotògraf Kolten Horner

Pel que fa al perfil dels surfistes de nou s'observa com semblen tots d'accés públic. Tal i com s'observa a la Figura 9, es mostra la mateixa informació que en un perfil de fotògraf: Nom, nacionalitat, likes, visites, followers, sessions, imatges i vídeos.

Per a seguir qualsevol perfil, o crear-ne un, la pàgina ens demana un registre senzill demanant només el correu i una contrasenya. A més, existeix la opció de realitzar el registre mitjançant Facebook. (Veure Figura 10)

Per a realitzar una compra també és necessari estar registrat a la pàgina. Al iniciar el procés de registre, (introduint un correu i contrasenya) la pàgina demana completar les dades escollint entre surfista i fotògraf, així com el nom, nacionalitat, una imatge de perfil (opcional), el gènere i la data de naixement. (Veure Figura 11)

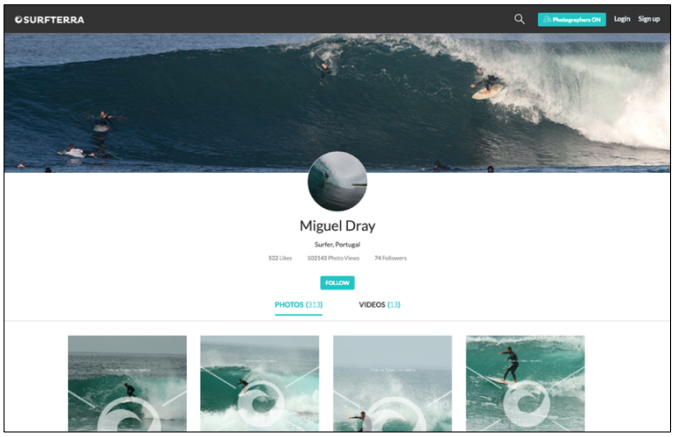


Figura 9 – Perfil surfista Miguel Dray

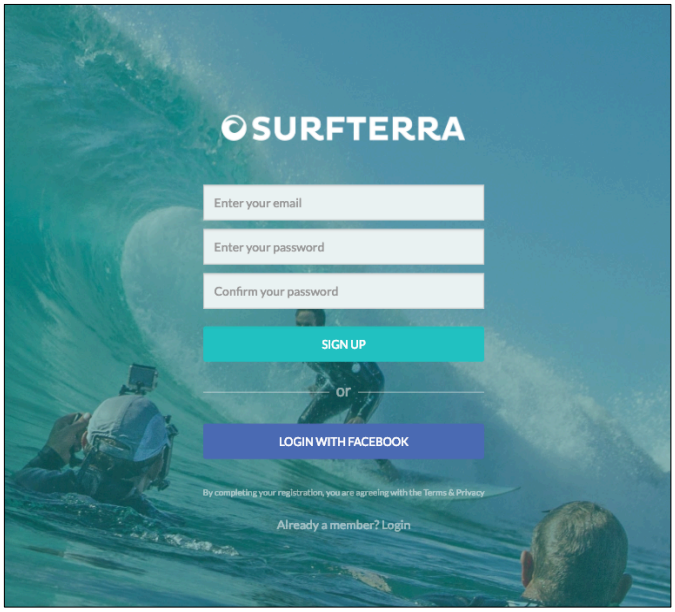


Figura 10 – Formulari registre Surfterra

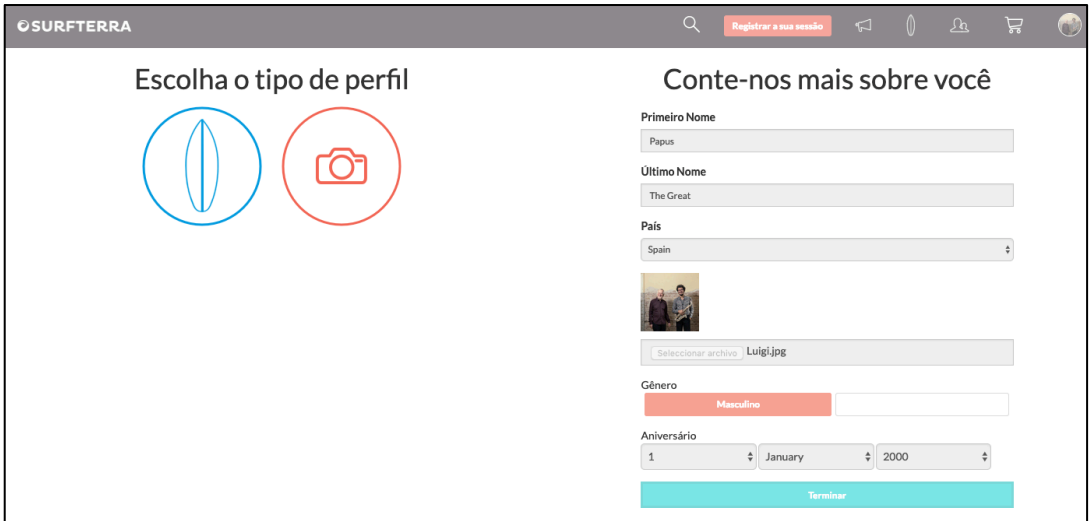


Figura 11 – Registre per a la creació d'un perfil

Un cop complert el formulari de registre, s'ha d'activar el compte mitjançant un enllaç rebut al correu introduït. (Veure Figura 12). Seguidament apareix la mateixa pàgina però s'observa la icona del nostre perfil conforme estem loguejats i es pot accedir a una pàgina de perfil on es troben les nostres dades, elements de configuració (Notificacions), un registre de compres i un llistat dels fotògrafs seguits (Figura 13). A més, apareix la opció de crear una sessió indicant la data i la localització on tindrà lloc i permetent, un cop creada, la pujada d'imatges i vídeos. (Veure Figura 14).

A la barra superior apareixen unes noves icones que actuen d'accés ràpid als fotògrafs seguits, sessions creades i carret de la compra. Tant si s'està registrat al portal o no, al *header* s'observa una barra de cerca que permet realitzar cerques específiques mitjançant filtre. (Veure Figura 15).

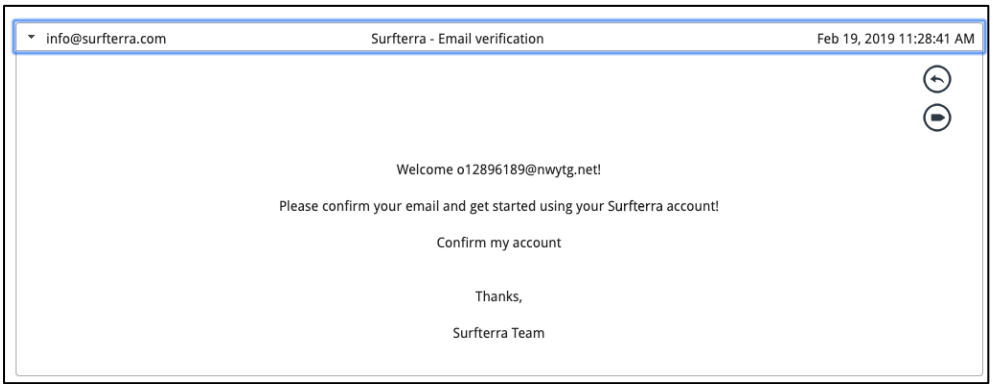


Figura 12 – Correu de confirmació de registre a Surfterra.com

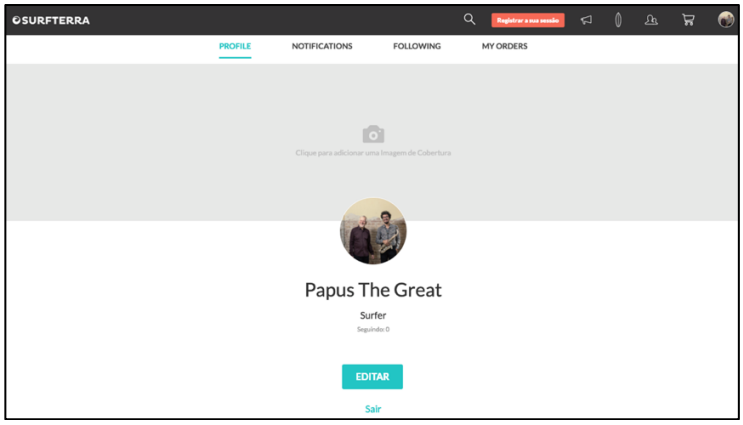


Figura 13 – Perfil d'usuari

Figura 15 - Cerca específica amb filtres (Surfista, fotògraf, localització, likes...)

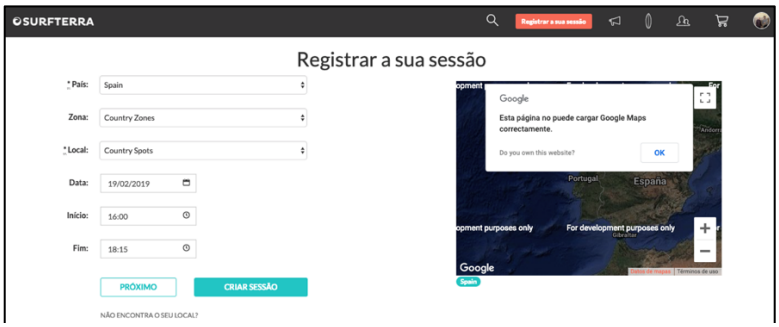


Figura 14 – Formulari de registre d'una sessió de surf

Pel que fa als fotògrafs el procés de registre és el mateix però seleccionant la opció de fotògraf. De la mateixa manera, s'haurà de validar el compte mitjançant un enllaç al correu introduït. Un cop validat es podrà accedir al perfil el qual disposa de les mateixes funcionalitats descrites anteriorment.

Per a penjar imatges o vídeos es necessari crear una sessió amb el formulari mostrat a la Figura 14. Un cop introduïdes les dades es poden penjar imatges i vídeos tot indicant el preu de venta i els surfistes que hi apareixen. En cas d'etiquetar a un surfista aquest rebrà una notificació que podrà veure al seu perfil. (Veure Figures 16 i 17).

Figura 16 – Opcions per a penjar imatges i vídeos

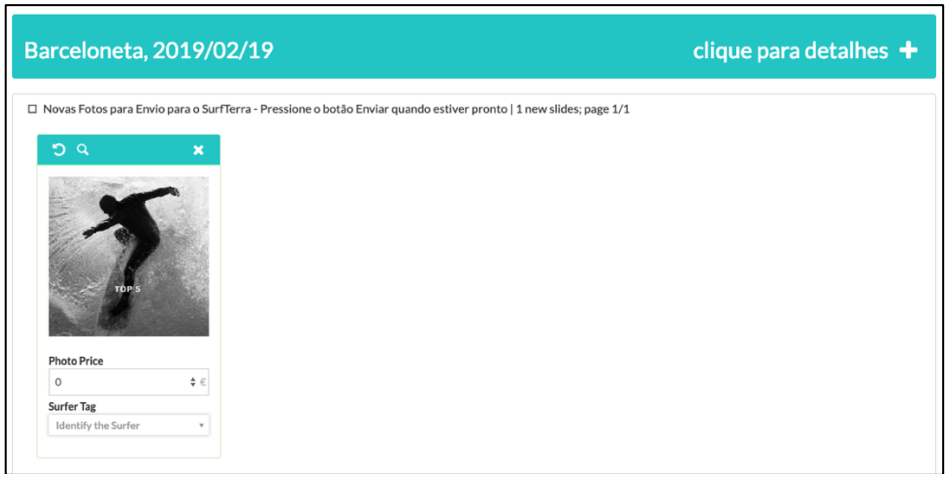


Figura 17 – Opcions de preu i etiquetat de surfistes al penjar una imatge

Yourbarrel

Es tracte d'una altre plataforma amb característiques molt similars que acaba de sortir aquest any 2019. Com s'explica a continuació, tot i ser menys atractiu que Surfterra.com a nivell de funcionalitats i disseny, es tracte d'un rival més fort, ja que actua a territori Espanyol i esta guanyant bastanta força.

El disseny de la plataforma és més simple amb poca combinació de colors. El primer detall de rellevància és que l'usuari no pot realitzar una cerca directament a la pàgina principal, sinó que ha de clicar sobre un botó per a accedir a una pàgina on apareix una barra de cerca. Aquesta barra permet la cerca per paraula clau (Localització o nom de fotògraf) però no permet l'aplicació d'altres filtres més específics. (Figura 18)

Al introduir una localització apareix un llistat de les imatges que coincideixen i com a novetat apareix un botó que permet consultar la previsió d'onades. Tot i que aquest servei pot ser força interessant el lloc on es troba el botó es clarament millorable ja que a la cerca d'imatges la previsió no és el que està cercant l'usuari.

Un altre aspecte millorable és el logotip i el canvi entre pàgines. Cada vegada que accedim a una pàgina nova apareix el logotip de la pàgina com a pantalla de càrrega. Al cap de consultar dues o més pàgines aquest efecte és més molest que agradable.

Les imatges disposen d'una marca d'aigua per a evitar descarregues i mostren informació de localització, data i un enllaç al perfil del fotògraf on podem trobar les ubicacions on acostuma a estar, les imatges penjades i la opció de sol·licitar una sessió privada. (Figura 19)

Pel que fa la compra d'imatges en aquest cas apareixen packs d'ofertes que permeten adquirir diverses imatges a un preu més econòmic. A més, a diferència de surfterra, el portal permet la compra d'imatges sense necessitat de registre. (Figura 20)

El formulari de registre és concís facilitant l'accés ràpid amb poca informació.

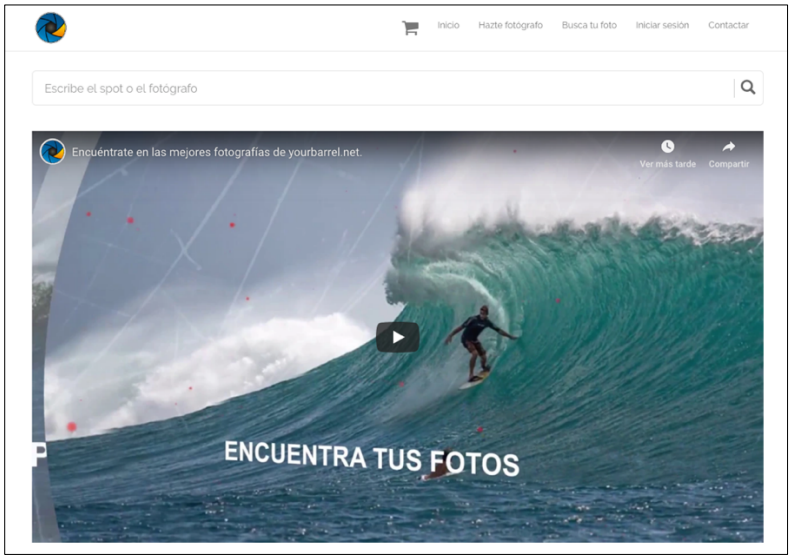


Figura 18 – Cerca d'imatges per Spot o Fotògraf

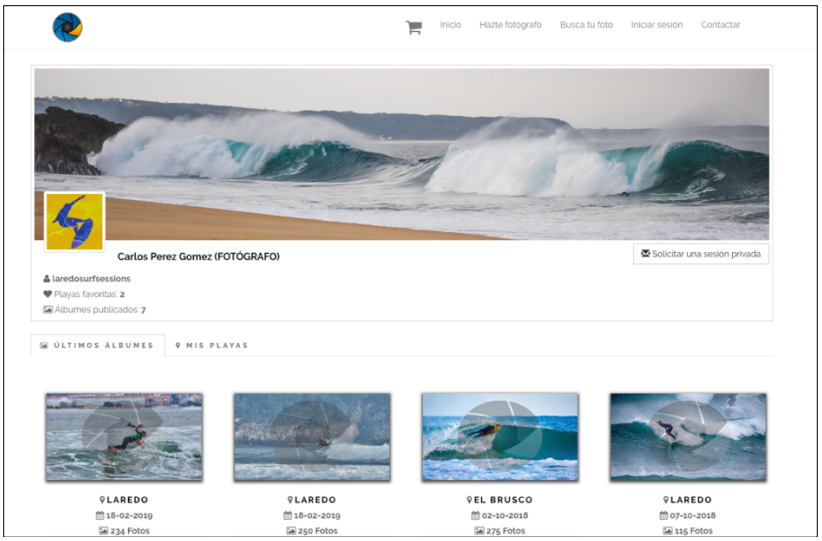


Figura 19 - Perfil amb opció a sessió privada

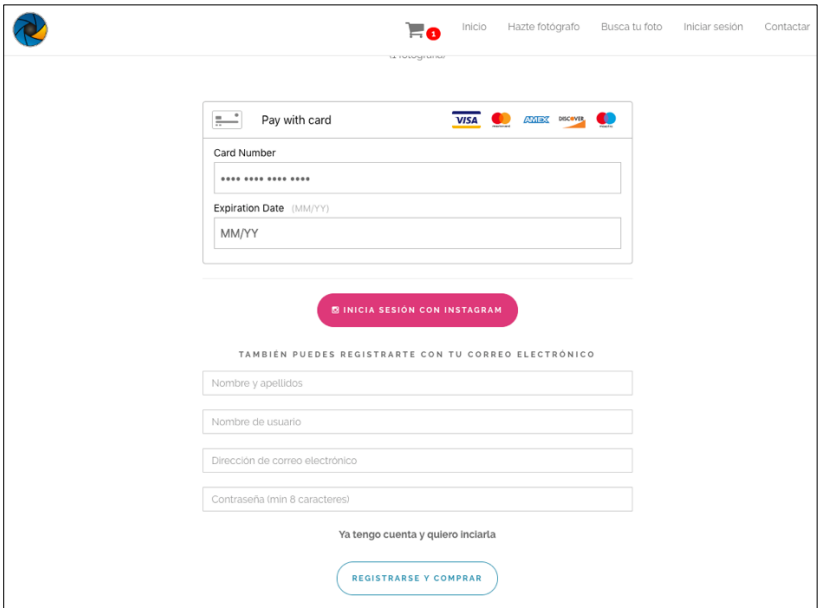


Figura 20 – Formulari de compra amb opció de registre

2.1 Conclusions de l'Estat de l'Art

Un cop estudiat l'estat de l'art es discuteix, en aquest apartat, els punts forts potencialment incorporables al projecte i els punts febles a evitar dels models similars vistos.

La conclusió més rellevant és que existeixen actualment projectes similars i que cada cop en surten més. Això indica que es tracte d'un mercat en augment on caldrà buscar una forma òptima d'aportar més valor i diferenciar-se de la competència.

A continuació s'analitzaran els punts forts dels competidors descrits a l'apartat anterior per tal d'incorporar-los al projecte aquí descrit, així com els febles per tal d'evitar-los.

Punts forts

- Accés lliure a la web però apartats restringits a membres registrats.
- *Responsive-design*
- Disseny de la pàgina: divisió d'apartats, colors contrastats, fàcil de llegir.
- Cerca per paraules clau i per apartats, a més es poden aplicar filtres per acotar els resultats. (Surfterra)
- Compra sense registre obligatori (Youbarrel).
- Possibilitat de registre mitjançant Instagram (Youbarrel)
- Possibilitat de contractar una sessió privada amb un fotògraf (Youbarrel)
- Compra per *packs* o ofertes (Youbarrel)
- Sistema de notificacions (Surfterra)
- Sistema de *Follow & Follow Back* (Surfterra)
- Opcions de privacitat (Surfterra)
- Marca d'aigua per a evitar descarregues i captures de les imatges
- Consultar previsió del temps (Youbarrel)
- Registre amb confirmació per correu
- Agent en línia per suport tècnic (Youbarrel)

Punts febles

- Pocs idiomes, dos en cas de Surfterra i només un en cas de Youbarrel
- No existeix una aplicació que faciliti encara més l'accés al sistema des de mòbils o tabletes.
- Aparició d'errors en el procés de registre i altres pàgines (Surfterra), Figura 21
- Mala disposició dels elements (Youbarrel)
- Icona de feedback de carrega per a l'usuari molest (Youbarrel)

A part de considerar els punts forts per a ser afegits, s'afegiran les següents funcions:

- Recomanacions per localitat de les millors zones de surf
- Estat actual del temps per localitat
- Recomanacions de escoles i campaments de surf amb filtres (Preu, localitat, etc.)
- Recomanacions per a lloguer de material amb filtres (Preu, localitat, etc.)

- Aplicació mòbil per Android i iPhone per a facilitar l'accés des de Smartphones i tabletes.
- Filtratge d'imatges per categoria esportiva: Surf, Kitesurf, Windsurf, etc.

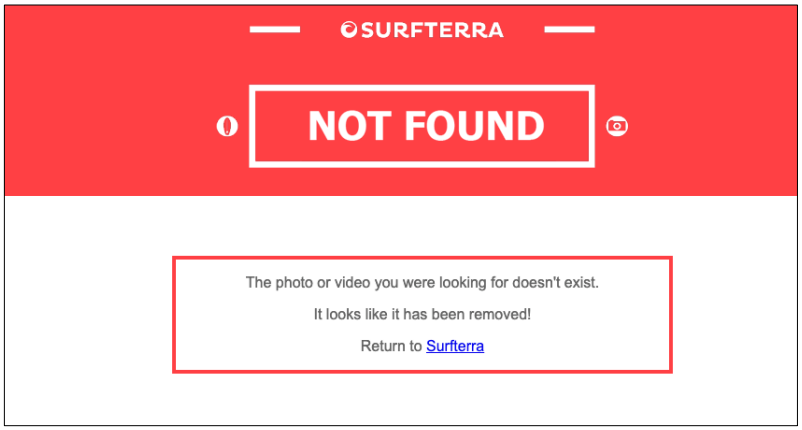


Figura 21 – Errors en alguns dels processos de registre de Surfterra

3 - Especificació de Requisits Software

Després d'analitzar l'estat de l'art i haver vist quins projectes similars hi ha al mercat es pot començar a definir el projecte de Shoreshots.

Es crearà una plataforma que pretenent convertir-se en marca referència sobre el món del surf i les seves variants. Els usuaris podran trobar informació referent a localitats, campus i campaments, meteorologia, descomptes, lloguer de material i últimes novetats.

El producte principal serà la venta d'imatges de surfistes. Per això, existirà el perfil de fotògraf el qual podrà pujar imatges a la venta. Els surfistes es podran cercar al portal mitjançant filtres (localització, data, etc.) i adquirir la imatge en línia o contactar amb el fotògraf. El portal tindrà un comportament de xarxa social permetent seguir a altres usuaris (Follow).

El sistema serà accessible des del portal web o des de les aplicacions mòbils, per tant, existirà un Web Service amb les funcionalitats necessàries definides. Les diferents plataformes accediran per a executar accions.

A partir d'aquest punt del document es fa un estudi de les millors maneres d'abordar projectes similars (Llenguatges, Connexions, Arquitectures, etc), seguidament, s'explica el procés escollit per al desenvolupament de Shoreshots, des del disseny fins a la implementació. Primer de tot, per a entendre millor aquest projecte es presenta a continuació l'ERS (Especificació de Requisits de Software), la primera pedra a l'hora de començar un projecte.

3.1 Especificacions de Requisits Software

A l'hora de participar en un projecte de tecnologia es pot donar el cas que el client no tingui molt clar què vol, o bé que ho vagi canviant al llarg del projecte. És per això que abans de començar amb el desenvolupament, s'especifiquen tots i cadascun dels seus aspectes i funcionalitats en un document anomenat ERS (Especificacions i Requisits Software).

L'ERS segueix el Standard IEEE std 830-1198 [5] i en ell es troben els diagrames de casos d'ús i activitat que descriuen a alt nivell el funcionament de l'aplicació. L'objectiu d'aquest document és que una persona amb pocs coneixements d'informàtica pugui entendre com funcionarà el sistema així com el d'establir un abast tancat del projecte i que el client no pugui anar modificant les característiques de l'aplicació sense control. Com que es tracte d'un document de llargada considerable es troba explicat al final d'aquesta memòria a l'apartat d'Annex.

4 Comparativa de Tecnologies

En aquest apartat s'estudien les diferents estratègies a l'hora de dissenyar i implementar un sistema distribuït mòbil/web. En aquest cas, el portal web i l'aplicació mòbil comparteixen funcionalitats, per tant, existeixen dues opcions:

- Dos sistemes independents amb funcionalitats replicades per cada sistema.
- Sistema distribuït amb un Web Service amb les funcionalitats definides. L'aplicació i el portal el consumeixen per a executar les funcions necessàries. Les funcionalitats només s'han de codificar una única vegada

Aquest apartat es centrarà en la segona opció. S'estudiaran diferents opcions per al disseny i implementació del Web Service, portal web i aplicacions mòbil així com mètodes de connexió entre els diferents sistemes dins d'una arquitectura client servidor.

4.1 Estructura Client-Servidor

L'arquitectura Client-Servidor és un model d'aplicació distribuïda on les tasques es reparteixen entre els proveïdors de serveis anomenats servidors i els demandants d'aquests serveis anomenats clients (Figura 22).

Així un client realitza peticions a un servidor qui executa una funció concreta i retorna una resposta al client. Com s'ha explicat anteriorment, per a aquest projecte, tindrem un Web Service que en aquesta arquitectura actua de Servidor i el portal web i les aplicacions el consumiran actuant com a clients.

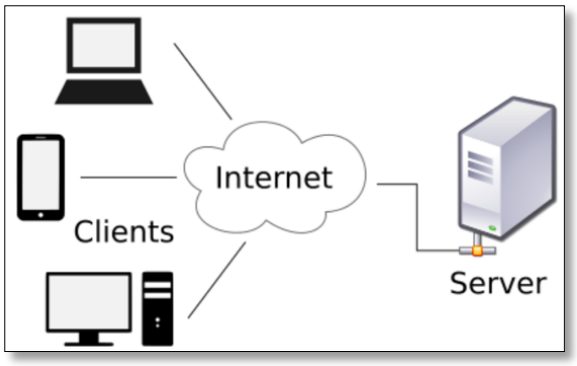


Figura 22 – Estructura Client – Servidor

Un cop definit el model d'arquitectura a estudiar, s'expliquen a continuació, diferents opcions per al disseny i implementació de cada un dels subsistemes. Com que alguns d'ells (Web Service i Web) comparteixen tecnologies d'ús, s'expliquen primer per a evitar repeticions.

4.2. Llenguatges

És un aspecte important a escollir a l'hora d'implementar un Web Service o portal/aplicació web és el llenguatge. A continuació es detallen alguns dels més usats i els seus punts forts i febles.

4.2.1 PHP (Hipertext Preprocessor)

Llenguatge de programació de propòsit general per backend. Dissenyat originalment per al desenvolupament web. Un dels primers llenguatges de backend en poder ser incrustats directament en un document HTML.

Compatible amb la gran majoria de servidors, sistemes operatius i plataformes. Actualment es troba instal·lat en més de 20 milions de portals web i més d'un milió de servidors. Permet als programadors desplegar aplicacions complexes amb una corba d'aprenentatge molt curta.

Existeixen diversos mòduls o extensions que permeten executar PHP de de línia de comandes igual que Pyhton o Perl (PHP-CLI), crear interfícies gràfiques d'usuari (PHP-GTK) i generar arxius PDF o Flash. També permet establir connexió amb diferents tipus de servidors de bases de dades com SQL, NoSQL, Oracle, DB2, MongoDB etc.

Permet una ràpida implementació Restful o Soap. Existeixen diversos Frameworks que faciliten la construcció. Alguns dels més rellevants són:

- **NuSoap Server**: NuSoap[27] és un ToolKit (Kit d'eines) basat en Soap per al desenvolupament de Web Services amb PHP. Està compost per una sèrie de classes que faran molt més fàcil el desenvolupament del servidor web.

- **Slim PHP**: És un micro Framework PHP que permet crear amb facilitat aplicacions web i APIs Restful basat en l'arquitectura model-vista-controlador. Els micro Frameworks són versions molt més lleugeres dels Frameworks, estan especialitzats en aplicacions web.

- **Laravel:** És un Framework PHP gratuït i de codi obert basat en l'arquitectura model-vista-controlador. És fàcilment configurable i extensible. Com a punt negatius destaquen la falta de comunitat de suport en comparació amb altres solucions i que és massa depenent de quèries a la base de dades, de vegades de forma excessiva provocant congestió. Laravel porta de sèrie integrat un CMS, CRM, un panell administratiu, etc. El que el fa molt complert per al disseny de portals i aplicacions web però potser massa extens per a crear APIs i Web Services petits.

- **Lumen:** Creat per Laravel, Lumen és un micro Framework PHP ràpid i lleuger. És una versió més lleugera que Laravel més orientada a la creació de APIs i microserveis (Procés independent que s'encarrega d'una única tasca. Ex: Cron-job)

- **Codeigniter:** Un altre Framework PHP, de codi obert i lleuger. Incorpora diverses eines i llibreries per a facilitar diverses funcionalitats tant per a la creació de Web Services com portals i aplicacions web.

- **CakePHP:** És un Framework per al ràpid desenvolupament de codi obert. L'objectiu principal es poder treballar de forma estructurada, ràpida i sense pèrdua de flexibilitat. Té una comunitat activa i segueix el patró MVC. Integra una sèrie d'eines per abstraure al programador de les tasques rutinàries i que es pugi centrar en la lògica (Bases de dades, Generació de codi, helpers i plantilles, etc.).

- **YiiFramework:** Està pensat per aplicacions web de gran escala. Basat en components d'altres prestacions permet la màxima reutilització en la programació accelerant el procés de desenvolupament. Porta eines per a la gestió de grans volums de tràfic i segueix el patró MVC. Sobresurt davant altres Frameworks en quant a la eficiència, gran quantitat de funcionalitats i extensa documentació. Ideal per a desenvolupar CMS, aplicacions web amb molt tràfic com portals i fòrums, e-commerce, etc.

Existeixen una gran varietat d'altres Frameworks i micro Frameworks PHP com Symphony, Epiphany, Frapi, Recess, Zend, Guzzle, Silex, etc. Aquests comparteixen moltes funcionalitats amb els altres Frameworks descrits anteriorment. Alguns estan més orientats al desenvolupament de grans aplicacions web mentre que altres són més lleugers pensats en crear APIs i petits Web Services. S'han destacat els explicats anteriorment per ser els més usats.

Segons l'article de 'Coders EYE' - 11 Best PHP Frameworks for Modern Web Developers in 2019 – l'ús d'aquests Frameworks en projectes d'aplicacions web es reparteix de la següent manera.

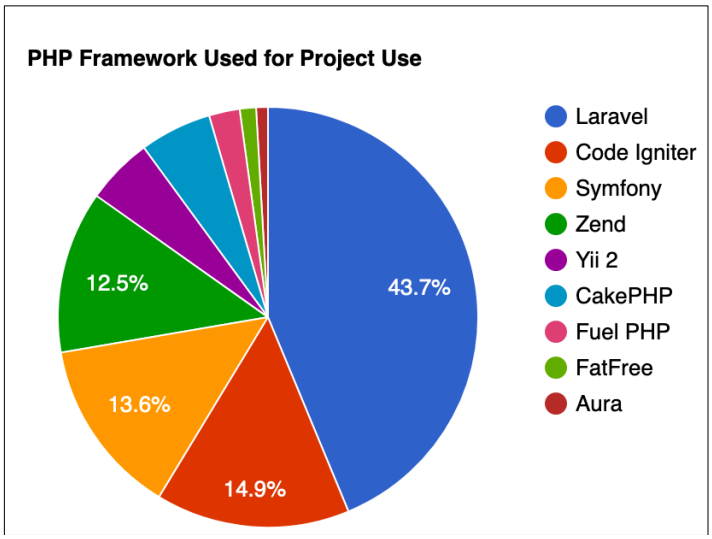


Figura 38 – Coders EYE ús dels Frameworks PHP en aplicacions Web

Com s'observa, Laravel és de lluny, el Framework més usat en PHP. Entre altres estudis, les dades s'extreuen de Google Trends per veure quins són els més cercats (Figura 39) on es torna a observar clarament com Laravel és el clar guanyador.

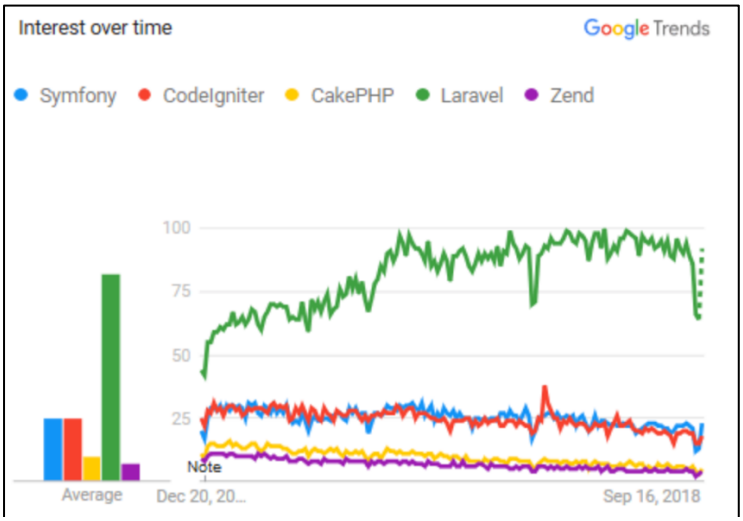


Figura 39 – Coders EYE anàlisis tendències amb Google Trends

IDES (Entorns de desenvolupament)

Com a entorn de programació per a la estructuració de fitxers i creació de contingut es poden usar una gran varietat de *IDES* (Integrated Development Enviorment) on destaquen: Netbeans, PHPStorm, Eclipse, Subime Text 3, Atom etc. La majoria d'aquests entorns de programació suporten altres llenguatges com Java.

Punts Forts	Punts Febles
• Pot accedir a més de 20 base de dades diferents incloent MySQL, Oracle, etc. • Suporta sessions amb ID's únics el que facilita la creació de botigues virtuals, etc. • La aplicació s'executa al servidor, no pot haver cap problema al client independentment del sistema o software que usi. • És fàcil d'aprendre amb una corba d'aprenentatge petita. • Conté eines i mecanismes per a crear sistemes complexos per a programadors avançats. • No és pot veure el *source code* com amb altres llenguatges com Javascript incrementant la seguretat. • Existeix una gran comunitat darrera per a resoldre errors i on trobar informació per al procés de creació. A més, hi ha disponibles fragments de codi per a la creació de funcionalitats específiques.	• El codi s'executa sempre al servidor podent provocar congestió en cas de molts clients fent sol·licituds concurrents. • Sistema d'errors. Tot i informar dels errors produïts, aquests molts cops són poc descriptius dificultant la tasca de correcció • És un dels llenguatges més problemàtics en quan als efectes negatius que te aplicar una actualització.

4.2.2 Java

És un llenguatge orientat a objectes dissenyat per a complir el concepte de *WORA (Write Once, Run Anywhere)* que consisteix en permetre que un codi programat i compilat es pugui executar en una nova màquina sense tornar a compilar.

És un dels llenguatges més usats des de 2012 sobretot en aplicacions web d'arquitectura Client-Servidor amb més de 10 milions d'instal·lacions. També és molt conegut per ser el llenguatge del sistema operatiu de mòbils Android.

A continuació es mostren alguns dels Frameworks més usats tant per a la implementació d'un Web Service com per a una aplicació o portal web.

- **JavaServer Faces (JSF)**: És un Framework de Java pensat per a dissenyar qualsevol cosa que es pugui fer amb llenguatge Java. És per això que no es tracta d'un Framework fàcil d'usar i requereix de coneixements i experiències prèvies. Forma part de l'entorn d'aplicacions Java EE (Enterprise Edition) i usa el llenguatge JSP per crear vistes HTML dinàmiques. És a dir, és un llenguatge similar a PHP que permet la inserció de codi en fitxer HTML com s'observa a la Figura 40.

```
<%@ page errorPage="myerror.jsp" %>
<%@ page import="com.foo.bar" %>
<html>
<head>
<%! int serverInstanceVariable = 1;%>
…
<%! int localStackBasedVariable = 1;%>
…
```

Figura 40 - Exemple inserció JSP a fitxer HTML

Com a punts forts cal destacar que es tracta d'un Framework suportat per Oracle per lo que disposa d'una extensa documentació. Disposa d'un gran conjunt d'eines i llibreries.

- **Spring:** És un Framework de codi obert per al desenvolupament d'aplicacions en Java. És un dels més antics però també està considerat un dels millors. Conté una sèrie de mòduls que permeten facilitar la creació de tasques específiques com per exemple el mòdul MVC (Model – View – Controller) per a la creació d'aplicacions Web. Una de les característiques més especials d'aquest Framework és el *contenidor d'inversió de control* el qual te la feina d'instanciar, inicialitzar i connectar els objectes de l'aplicació.

Els objectes gestionats per aquest sistema s'anomenen *beans* i són del tipus *POJO* (Plain Old Java Object, Objectes que no depenen d'un Framework en especial). Existeix molta documentació per a crear pràcticament qualsevol cosa així com una molt bona comunitat darrera. Com a punt "negatiu" es pot dir que es un Framework complex que requereix de coneixements previs i te una corba d'aprenentatge gran. Tot i això, també es una bona opció per a la creació de Servidors Web REST degut a la extensa documentació i a les múltiples eines que ho faciliten.

Spring incorpora SpringWS el qual facilita la creació de Servidors Web usant SOAP.

- **Struts:** En la seva segona versió, és un Framework de codi obert pensat per a la creació d'aplicacions web i basat en el model – vista – controlador. Permet implementacions ràpides i tasques de testing de forma còmode i ràpida degut a que es capaç de crear nous segments de codi.

  Dit això, no es un Framework molt flexible, si s'usa per a un projecte s'han d'acceptar les seves regles. És a dir, per als programadors que tenen certes metodologies per a la implementació d'algunes tasques no és la millor opció ja que Struts marca les pautes a la seva manera. És extensible mitjançant l'ús de plugins per exemple, suportar JSON, AJAX i REST per a poder crear Web Services.

  De la mateixa manera que Spring, Struts es pot ampliar amb llibreries que faciliten la creació de Servidors Web basats en SOAP

- **Apache Wicked:** Amb un gran suport per part d'Apache, Wicket és un Framework més lleuger dissenyat per a la creació d'aplicacions web simples. És de codi obert i de la banda de servidor però fàcilment integrable amb HTML per a la creació de vistes. Té una amplia documentació i es un Framework ideal per a la creació de pàgines i aplicacions elegants.

- **Vert.x:** Es tracte d'un Framework molt versàtil que suporta diversos llenguatges on el més comú és Java. Una de les seves característiques més rellevants és la versatilitat de configuració on es pot escollir quines llibreries usar i quines no. S'executa sobre JVM (Java Virtual Machine) i permet testejar el codi i escalar-lo de forma immediata.

- **Play:** Es tracte d'un Framework molt simple i fàcil d'usar gràcies a la seva interfície gràfica. Està dissenyat per a la creació d'aplicacions modernes on l'aspecte de ser simple de cara als usuaris finals és molt important. Aquest Framework també s'executa sobre JVM permetent escollir els recursos de l'ordinador a usar (RAM, CPU).

  Molts programadors que han usat aquest Framework asseguren que ha augmentat la seva productivitat degut a la senzillesa d'ús. Tot i que amb la versió actual no fan falta gaires millores, un dels aspectes negatius de play és les poques actualitzacions.

- **Grails:** És un Framework per al desenvolupament d'aplicacions web sobre el llenguatge Groovy que esta basat en Java. Presenta un entorn de treball estandarditzat ocultant moltes parts de configuració al programador i aconseguint així alta productivitat. Al seguir la documentació els aspectes de configuració són pràcticament inexistents i ràpidament es passa a construir la primera aplicació el que pot ocupar 2-3 hores.

  Per tant, podríem dir que la corba d'aprenentatge és més petita respecte altres Frameworks. Disposa d'una amplia documentació i més de 900 plugins per a la realització de tasques específiques, alguns d'ells per exemple, per a la creació de Web Services. Es compatibles amb els IDE's més usats de Java com Eclipse o Textmate els quals s'explicaran més endavant. Com aspecte potser més negatiu està el fet de que cal usar la màquina virtual de Microsoft .Net (Runtime Language)

- **Apache Axis 2:** És un redisseny total del Framework basat en SOAP Apache Axis. Existeix una implementació en Java i una en C. Incorpora totes les eines necessàries per a construir un Web Service basat en XML com suport WSDL 1.0 i 2.0

- **JSF:** Java Server Faces, tot i no ser dels millors Frameworks en Java per al desenvolupament ràpid, es molt fàcil d'usar gràcies, entre altres, per una molt bona documentació per part d'Oracle. Te llibreries i eines potents per a funcionalitats específiques i complexes. Tot i ser senzill d'usar requereix de coneixements Java bastant sòlids.

- **Vaadin:** Usa Google Web Toolkit per al renderitzat de la pàgina visual i sha convertit en una opció molt escollida per al desenvolupament d'aplicacions web. Permet el patró MVC o MVP que s'expliquen més endavant. També incorpora funcionalitats de Drag & Drop per a la creació de pàgines.

Existeixen molts mes Frameworks amb aquestes característiques o amb funcions més específiques com Google Web Kit amb eines per a modificar el *frontend* d'una aplicació web, Hibernate més pensat per a la connexió amb diverses bases de dades independentment del tipus.

Per a tasques tant concretes com la creació d'un Web Service existeixen una sèrie de micro Frameworks més pensats per aquestes tasques, sobretot per a servidors REST:

- **Dropwizard**: Posa a disposició una sèrie de llibreries per a funcionalitats específiques com pot ser la creació de servidors REST. Es perfecte per a desenvolupaments ràpids, modular, ràpid, amb una gran comunitat darrera i eines de monitorització.

  Usa Jetty per HTTP (Servidor HTTP basat en Java pensat per a la creació de servidors web senzills), Jersey per a REST (Framework de Java per a la creació de servidors REST) i Jackson per a JSON (Llibreria en Java per a la conversió de classes a text en format JSON). Com a punt negatiu destaca el fet que al usar moltes llibreries de tercers, també incorpora els seus errors (*bugs)*

- **Light-Rest-4J – Light Java:** Aquest Framework, en ell mateix és una API REST sobre light-4j, que és un Framework de micro-serveis cloud molt lleuger. És un dels Frameworks més ràpids, si no el que més, per al desenvolupament de servidors REST. Està dissenyat pensant en la escalabilitat dels projectes dissenyats, s'integra fàcilment amb altres llibreries i Frameworks i és modular, és a dir, podem afegir i treure plugins fins a fer-ho el més petit necessari per als nostres requeriments. Tot i aquests diversos punts forts, no te masses seguidors comparat amb altres serveis similars. Un del possibles motius és la escassa documentació.

- **Ninja:** És similar a Dropwizard, disposa de tot el necessari per al desenvolupament d'un servidor REST. Permet crear un projecte sense masses aspectes de configuració i importar-lo des de IDE's compatibles. Alguns dels seus aspectes positius són la modularitat, la rapidesa, la renderització en XML, JSON i HTML i la compatibilitat amb altres llibreries. No és gaire usat en comparació amb altres Frameworks similars i per això la documentació i suport online no són tant bons.

- **Resteasy:** És una solució basada en JBoss (Servidor d'aplicacions JavaEE de codi obert, al estar en Java es pot usar en qualsevol sistema operatiu. Actualment s'anomena WildFly i va ser adquirit per Red Hat l'abril del 2006) que integra una sèrie de diversos Frameworks i llibreries per a la creació de servidors REST. És un dels Frameworks més antics amb una comunitat

considerable darrera. Té un excel·lent suport de seguretat (OAuth 2.0) usa Spring MVC descrit anteriorment del qual potser depèn massa. Usa JAX-RS 2.0 que és una especificació de Java pensada per a la creació de Web Services. Per altre banda, no disposa de masses eines de testing i té una documentació poc elaborada.

- **Restlet:** Un dels primers Frameworks de Java amb aquestes característiques per a la creació de Servidors REST. Ofereix diverses extensions per a encaixar el millor possible amb les necessitats dels programadors. Usa la API JAX-RS que proporciona suport per a la creació de Servidors REST. Manté suport amb altres llibreries i segueix actualment en desenvolupament d'actualitzacions per a incorporar millores. Com a punts negatius necessita d'una corba d'aprenentatge força gran, no és gaire famós avui en dia, en part, gràcies a Jersey i Play i té una petita comunitat bastant tancada darrera.

A continuació es mostra l'estudi de Rebellabs, el qual manté un index de popularitat entre els Frameworks de Java basat, entre altres coses en comparacions amb Stackoverflow, Github, Linkedin i Google search. A la Figura 41 es mostren els 11 primers Frameworks amb millor índex de popularitat.

Rank	Framework	Popularity
1	Spring mvc	28.82
2	JSF	15.2
3	Spring Boot	13.35
4	GWT	7.74
5	Grails	6.35
6	Struts	5.4
7	Dropwizard	4.9
8	Play framework	3.26
9	JHipster	2.49
10	jax-rs	2.44
11	Vaadin	2.15

Figura 41 – Rebellabs índex de popularitat

Rebellabs permet, mitjançant una gràfica interactiva, anar comparant l'índex al llarg del temps tal i com s'observa a la Figura 42. A les dues Figures de Rebellabs s'observa com Spring és de lluny el Framework més usat i amb millor evolució.

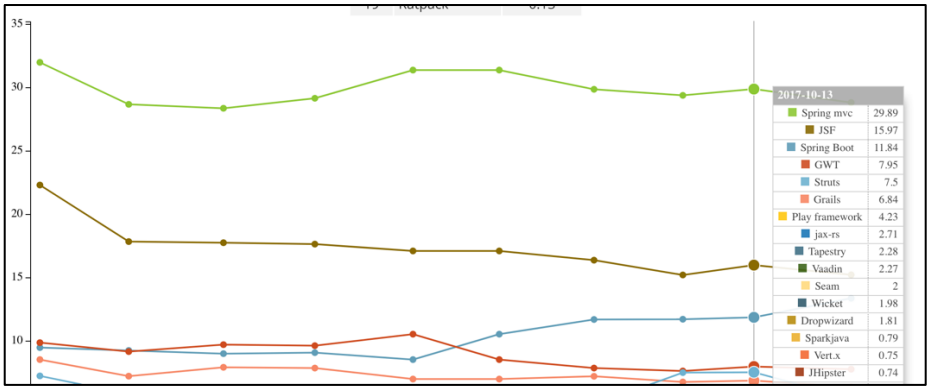


Figura 42 – Rebellabs evolució de l'índex al llarg del temps

IDES (Entorns de desenvolupament)

Els entorns de desenvolupament de propòsit general més famosos per Java ja s'han nombrat anteriorment i són NetBeans i Eclipse. També són coneguts altres com JSource o JDeveloper

Existeixen altres IDEs de Java més específics com poden ser Android Studio i IntelliJ IDEA per al desenvolupament d'aplicacions Android, BlueJ o DrJava amb propòsits més educacionals o JEdit amb característiques específiques d'editor de text.

Punts Forts	Punts Febles
• Multiplataforma. No cal conèixer la màquina on s'executarà el codi a priori. • Gran quantitat de documentació i informació • Gran conjunt d'eines i llibreries extens. • Orientat a objectes. • El programador no s'encarrega d'alliberar memòria sinó que per aquesta tasca està el *Grabage Collector.* • Relativament senzill d'aprendre respecte a altres llengautges. • Gestió d'errors amb excepcions.	• El rendiment pot ser lleugerament inferior al de altres llenguatges degut a que es tracta d'un llenguatge interpretat. • Només es pot executar un codi Java si es disposa d'un entorn JVM (Java Virtual Machine) • Per a principiants, l'orientació a objectes no és la filosofia més adient. • Alguns programadors troben la sintaxi més complexa que altres llenguatges com Python.

4.2.3 Python

És un llenguatge de programació interpretat, és a dir que no requereix de compilació ja que s'interpreta en temps real el que en la majoria de casos deriva en una millor eficiència. A més, els llenguatges interpretats, com Java descrit anteriorment, tenen la avantatge de ser multiplataforma, és a dir que no cal pensar on s'executarà el codi perquè serà compatible independentment de la màquina que hi hagi darrera.

Python és multiparadigma, és a dir que es pot programar de forma imperativa (Canvis d'estat), orientat a objectes (S'instancien objectes amb diversos atributs i funcionalitats que treballen sobre les dades) o orientat a funcions (Canvis mitjançant funcions matemàtiques).

Un dels objectius de Python és obtenir codis llegibles fàcils de comprendre mitjançant una sintaxi simple i donant facilitats com per exemple el tipatge dinàmic el qual permet que una mateixa variable tingui diferent tipus de valors en diferents moments (Enter, Caràcter, etc.)

A la Figura 43 es mostra la evolució de Python on s'observa com la seva popularitat creix sense parar convertint-se en un dels llenguatges més usats. Daffodil – Growth of major programming languages.

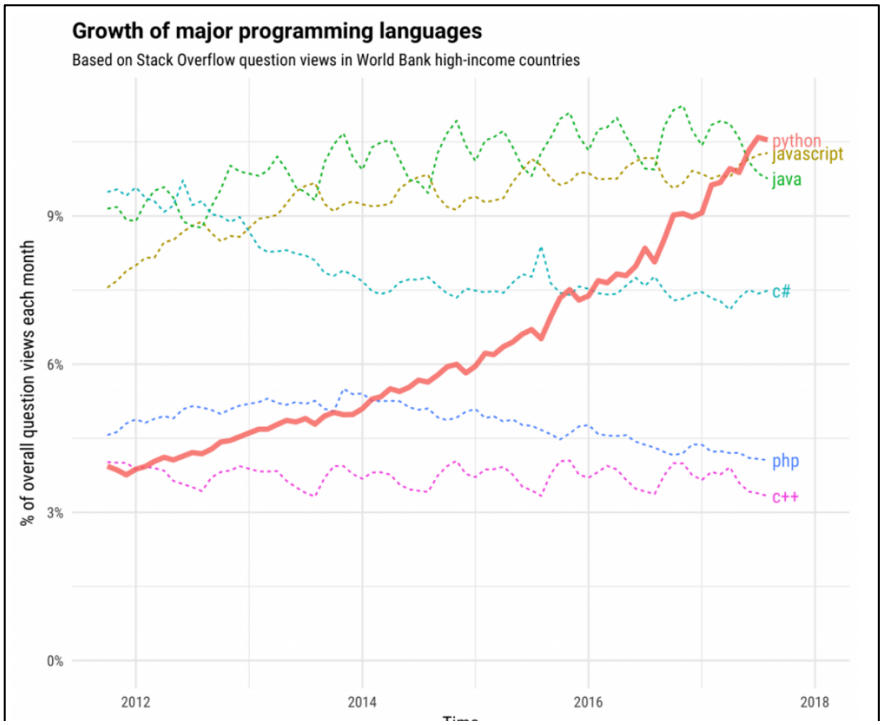


Figura 43 – Daffodil – Growth of major programin languages.

A més d'aquesta forta evolució, un dels punts forts de Python és la seva versatilitat, i es que tot i començar amb molta força en el mon d'anàlisi de dades, Python s'ha anat expandint a altres sectors de programació tal i com ens mostra també l'estudi de Daffodil on s'observa l'ús del llenguatge en àmbits diferents de programació.

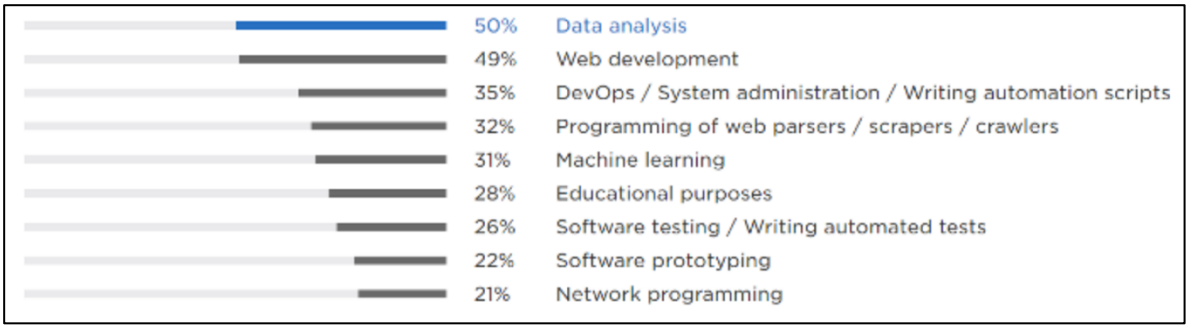


Figura 44 – Daffodil Verstilitat de Python

A continuació es mostren alguns dels Frameworks més usats per al desenvolupament de portals i aplicacions web així com Web Services:

- **Django**: És per molts el millor Framework web basat en Python. Permet construir aplicacions potents de forma ràpida amb un gran conjunt d'eines integrades per a gestionar sessions, autenticació, URL Routing etc. Segueix el patró Model – View – Template i és de codi obert. Consta d'una documentació extensa així com una gran comunitat darrera.

  De forma modular permet gestionar les llibreries i extensions disponibles, entre elles, existeixen algunes per al desenvolupament de Servidors REST com Django REST o Spyne per Servidors Soap.

  És compatible amb nombroses bases de dades com PostgreSQL, MySQL, Oracle o SQLite de forma que un mateix codi funciona amb diferents bases de dades. Django és usat per moltes companyies grans com Pinterest, Instagram, Bitbucket, Mozilla o el Washington Times.

  Django també te característiques per a emfatitzar la seguretat com formularis amb CSRF Tokens o encriptació de sessions. Els CSRF Tokens són arguments que s'afegeixen a la url per a evitar peticions malicioses a traves de la url. Per exemple, si imaginem que estem loguejats a la web del banc, tenim la sessió oberta i accedim a una altre pàgina maliciosa que intenta aprofitar que estem loguejats al banc per mitjançant la url realitzar una transferència. És necessari tenir coneixements previs però si sabem que per exemple el banc realitza transferències amb la url:

http://www.elmeubanc.com/transferencia?to=123456;cantitat=10000;

Al estar en sessió el banc agafa la sol·licitud com a vàlida. Amb un token CSRF s'afegeix un argument al final que fa impossible esbrinar la url.

http://www.elmeubanc.com/transferencia?to=123456;cantitat=10000;token=3141592653589793238462643383279502884197 1

- **Pyramid**: Inicialment amb el nom de Pylons, és un Framework Python que s'esforça per ser minimalista, ràpid i fiable i un dels primers compatibles amb Python 3. És ideal per a aplicacions web grans com CMS's, però també es pot adaptar sense problemes a projectes més petits. te una documentació excel·lent i la seva comunitat està creixent força.

  També és una bona opció per al disseny de Servidors REST degut a funcionalitats que te per defecte com mapejar URL's a codi com Django. És també força conegut per la seva seguretat i ha estat usat per companyies importants com Mozilla, Yelp i Dropbox.

- **Bottle:** És un micro Framework de Python molt més lleuger que els anteriors. Originalment desenvolupat per a la creació de APIs REST. Està pensat per a aplicacions molt petites intentant executar-les en un únic fitxer. Tot i això te eines i funcionalitats per a construir interfícies web atractives. Netflix és una de les companyies que ha usat Bottle per al disseny de interfícies web.

- **Flask:** És un altre microframework que es va crear com una broma el dia d'April Fools i que va derivar en un Framework d'un únic arxiu com Bottle. Intenta ser el màxim simple possible però es pot anar augmentant funcionalitats mitjançant mòduls. Principalment és ideal per a principiants que busquen aprendre a programar o per petits prototips o aplicacions petites.

- **Web2py:** És un Framework de codi obert per el desenvolupament d'aplicacions web. Segueix el patró de disseny model – Vista – Controlador, és bastant complert amb bones opcions de debugger, edició de codi, desplegament i testing. Una de les característiques principals és un sistema de tiquets que es generen quan apareix un error, d'aquesta manera el programador pot seguir les indicacions del tiquet per situar i solventar l'error en qüestió. Permet incorporar llibreries externes però ja porta diverses incorporades per la gestió de HTTP requests, sessions i cookies entre altres.

Existeixen altres Frameworks amb característiques similars com CherryPy, Grok, TurboGears, Falcon i Tornado. Aquest últim, està dissenyat per a aplicacions web molt grans que han de gestionar moltes connexions simultànies.

Per al desenvolupament de Web Services existeixen una gran varietat de llibreries majoritàriament compatibles amb els Frameworks descrits anteriorment. A continuació s'enumeren una sèrie de llibreries per al desenvolupament SOAP: ZSI, soaplib, suds, pysimplesoap, osa, Ladon Zeep, etc.

Per altres desenvolupaments més particulars com JSON-RPC o JSON-WSP es poden usar les següents llibreries Python: python-json-rpc, JsonUtils, ladon ,pfacka/jsonwsp

Entre tots els Frameworks descrits destaca per la seva popularitat d'ús Django. Més endavant es mostrarà una compartiva general entre llenguatges i Frameworks on s'observa com Django és dels més usats.

IDES (Entorns de Desenvolupament)

A continuació es mostren els entorns de desenvolupament Python més coneguts i usats. Alguns d'ells tenen diferents versions, una de pagament més professional i una versió de comunitat gratuïta.

Pycharm és un exemple de IDE amb dues versions. La versió de comunitat és més que potent incorporant eines per a debugar, testing, desplegar sistemes i accessos a la base de dades. És dels mateixos creadors d'altres IDES potents descrits anteriorment com PHPStorm.

AWS Cloud9 és un IDE basat en cloud desenvolupat per Amazon. Incorpora eines per al desenvolupament, desplegament i testeig de projectes. Suporta diversos llenguatges i permet el treball en equip de forma fàcil.

Komodo és un altre IDE multi llenguatge, com a característiques destacables destaca la tecnologia d'autocompletat i refactorització. A més suporta controls de versions com Git, Mercurial o Subversion per al treball en equip.

Altres IDES com Codenvy, KDevelop, Anjunta o Wing Python son compatibles amb Python i tenen altes prestacions per al desenvolupament d'un projecte. Per altre banda, alguns dels IDES descrits en llenguatges anteriors també es poden usar amb Python mitjançant extensions com Eclipse o NetBeans.

A continuació es mostren alguns dels punts forts i febles de Python:

Punts Forts	Punts Febles
• Té una sintaxi simple i fàcil d'entendre. • Llenguatge interpretat, multiplataforma • Flexible, Multiparadigma • Tipatge dinàmic, una variable pot prendre diversos tipus de valors en diferents moments. • Fàcil d'integrar. Disposa de mòduls per a poder integrar-se amb altres plataformes i llenguatges. • Afavoreix la productivitat. • Cobreix un ampli sector de recursos de forma nativa i mitjançant llibreries externes (JSON, XML, HTML, correu electrònic, operacions amb cadenes, etc.)	• El tipatge dinàmic en casos **molt** concrets provoca errors. • És un llenguatge més lent en execució que altres. Ho compensa sent més ràpid per a desenvolupar.

4.2.4 Ruby

Ruby és un llenguatge interpretat orientat a objectes i de codi obert enfocat a la simplicitat i la productivitat amb una sintaxi simple inspirada en Python. Entre les seves característiques més generals estan el tipatge dinàmic, la gestió de excepcions, *garbage collector*, amplies llibreries, flexibilitat etc.

A Ruby, tots els tipus de dades son objectes, inclosos els tipus primitius com enters i booleans. Les variables sempre son referències a objectes. És multiparadigme, és a dir que suporta diversos dissenys de programació com la imperativa on es defineixes en funcions i variables fora d'una classe fent-les part de l'objecte arrel, orientació a objectes o orientat a funcions.

Va ser inventat pel programador Japonès Yukihiro Matsumoto amb la filosofia que fes disfrutar als programadors per la seva senzillesa. Avui en dia és una molt bona opció per al desenvolupament web.

Tot i que Ruby es manté com una molt bona aposta, la seva popularitat ha deixat de créixer per a començar a baixar en aquests últims anys. Segons el Blog Medium i l'article de Joel Blum, la tendència de Ruby i el seu màxim exponent, Ruby on Rails és clarament negativa. A la Figura següent es mostra aquesta evolució.

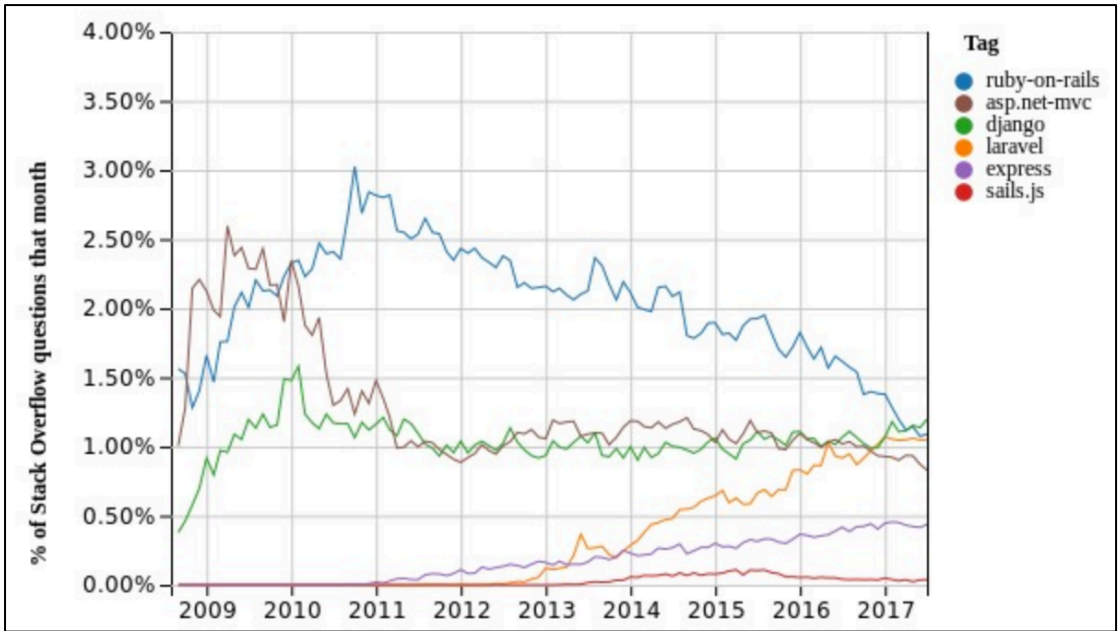


Figura 45 – Medium – Joel Blum – Evolució de la popularitat de Ruby

Tot i aquesta gràfica, Ruby on rails segueix sent una de les opcions més usades per al desenvolupament web.

A continuació es mostren les opcions de Frameworks més recomanables per a començar un projecte d'aplicació web amb Ruby:

- **Ruby on Rails:** És sens dubte el Framework més famós de Ruby. Incorpora diverses característiques per a facilitar el desenvolupament web tant a la banda de client com a la de servidor. Permet desenvolupar serveis web REST i SOAP així com connexions amb diversos models de base de dades.

  Permet als desenvolupadors crear interfícies d'usuari combinant CSS, HTML, JavaScript i XML. Permet agregar funcionalitats sense escriure codi addicional mitjançant l'ús de la varietat de *gemes* (llibreries) que disposa. RubyGems és el *Package Manager* que permet la gestió d'instal·lació de gemes en un format propi anomenat *gem.*

Segueix el patró de disseny Model – Vista – Controlador i intenta obtenir la màxima productivitat del programador amb un mínim de configuració i seguint la simplicitat de Ruby per a fer més escrivint menys.

Un altre dels principis més característics de Ruby on Rails és el DRY (Don't repeat yourself) que permet fer les definicions un únic cop. Per exemple, Ruby pot esbrinar el nom de les columnes d'una classe per la base de dades de manera que definir-ho també a codi seria redundant.

També incorpora una convenció sobre la configuració de manera que per exemple si hi ha una classe *User*, per defecte la taula corresponent a la base de dades serà *users*. No haurem d'afegir més codi a no ser que vulguem modificar aquesta configuració per defecte i renombrar la taula.

- **Sinatra:** És un Framework per al desenvolupament d'aplicacions web de codi obert i un llenguatge de domini específic (DSL – Domain-Specific-Language). No és compatible amb el patró MVC però és usat per als desenvolupadors per tenir un marc simple i lleuger per a codificar de forma ràpida aplicacions en Ruby.

- **Padrino:** Està basat en Sinatra. És un Framework molt lleuger i simple però que incorpora eines i mecanismes per a simplificar el disseny i implementació d'aplicacions web de diverses mides. Permet un desenvolupament ràpid amb una interfície de creació d'administrador com Django, inclou ORM el qual transforma taules de la base de dades a objectes per a facilitar l'accés a les dades al programador, inclou autenticació, creació de templates. Incorpora llibreries de testing i de connexió amb la base de dades.

- **Cuba:** És tracte d'un micro Framework per a Ruby. És lleuger i molt ràpid i permet desenvolupar aplicacions web de forma ràpida. Disposa d'eines bàsiques molt útils per al desenvolupament però es poden ampliar amb altres llibreries per al desenvolupament de templates o testing.

- **Roda:** Usa un arbre d'enrutament per a complir la propietat de DRY (On les definicions només cal fer-les un cop) i així mantenir el codi més simple possible. És més ràpid que altres Frameworks degut a funcions intel·ligents de caché per a guardar dades internes. Està força enfocat a la seguretat amb formularis amb Tokens CSRF com Django i sessions encriptades. Porta diversos plugins incorporats de sèrie però extensibles i suporta el patró MVC.

- **Pakyow:** Permet implementar la interfície d'usuari sense preocupar-se de la lògica de darrera, és a dir, segueix el procés de disseny View-First. Aquest procés actualitza la vista amb les dades proporcionades sense moure el codi a la banda de client.

Existeixen altres Frameworks amb característiques similars com Trailblazer, Scorched o Goliath.

IDES

A continuació es mostren alguns dels entorns de programació més usats a ruby.

RubyMine, de la companyia JetBrains els mateixos creadors que PyCharm i PhpStorm descrits anteriorment així com molts d'altres. Permet produir codi d'alta qualitat de forma eficient. Suporta Ruby on Rails, Javascript, CSS, etc. Disposa d'una amplia documentació i te funcionalitats de correcció de sintaxi, anàlisis d'errors i autocompletació de codi.

Els editors de text Vim i Emacs es poden configurar com a entorns de desenvolupament per a Ruby aportant noves funcionalitats que els converteixen en IDES perfectament complerts tot i que inferiors a les opcions que es poden trobar a RubyMine.

Komodo, que ja s'ha explicat anteriorment, també s'usa amb molta freqüència com a entorn de desenvolupament per a Ruby. És multiplataforma i tal i com s'ha explicat abans, disposa de diverses funcionalitats per a la programació còmoda i eficient.

A continuació es mostren els punts forts i febles de Ruby com a llenguatge de programació, com es pot observar, molts dels punts forts són compartits amb Python.

Punts Forts	Punts Fèbles
• Desenvolupament ràpid. Té una sintaxi simple i fàcil d'entendre que permet crear aplicacions de forma ràpida. • Llenguatge interpretat, multiplataforma • Tipatge dinàmic, una variable pot prendre diversos tipus de valors en diferents moments. • Afavoreix la productivitat.	• Els Frameworks més rellevants de Ruby com Ruby On Rails consumeixen bastants recursos (CPU, RAM) en comparació amb altres tecnologies similars. • Al ser interpretat és més lent que altres llenguatges. • Aplicacions multifil amb ruby tenen un rendiment millorable degut a que ruby usa GIL (Global Interpreter Lock) que nomes permet executar un *thread* a la vegada encara que ens trobem en un sistema amb multiprocessador.

4.2.5 - Go

Go o Golang és un llenguatge compilat inspirat en la sintaxi de C desenvolupat per Google l'any 2009 i disponible per les diferents plataformes (Windows, Mac OS X, Linux). Com a característiques suporta la programació imperativa i orientada a objecte si disposa de *Garbage Collector* per a gestionar l'alliberament de memòria de forma transparent al programador.

Amb una sintaxi similar a C, usa tipatge estàtic on una variable ha de tenir el mateix tipus de valor després de ser declarada. Per altre banda, te molts aspectes similars a Python en quan a senzillesa. Altres llenguatges com Java o C++ són molt més voluminosos i pesats.

Go busca simplicitat amb una sintaxi clara i concisa diferenciant-se de C en molts aspectes com per exemple la declaració de variables o l'ús opcional del punt i coma al final de línia. Go no usa excepcions per a no afegir complexitat al llenguatge ni aritmètica de punters per raons de seguretat i per simplificar la tasca del *garbage collector*.

Go permet la programació orientada a objectes però no admet la construcció de jerarquies, és a dir no existeix la herència entre objectes.

En quant a popularitat, segons el portal mèdium i l'article de Edvin Dizdarevic - GoLang, The Next Language to Learn for Developers – Go ha aconseguit escalar fins al top 5 dels llenguatges més estimats per als programadors segons dades i enquestes a Stackoverflow, Google trends, etc.

També s'ha convertit en un dels llenguatge de creixement més ràpid a Github. I segons les enquestes, està primer en quant al llenguatge que els desenvolupadors tenen en ment aprendre als pròxim anys (Figura 46)

Figura 46 – Medium – Edvin Didzarevic – Llenguatges a aprendre als pròxims anys

A continuació es mostren alguns dels Frameworks més usats per a Go tant per al desenvolupament web com per a la construcció de Web Services REST o Soap.

- **Revel:** És un Framework d'alt rendiment per a GO que inclou de fàbrica funcionalitats complexes per a no necessitar llibreries externes, és flexible permetent diversos models de configuració segons l'escenari a construir. Incorpora funcionalitats com un sistema de recompilació del codi cada vegada que es guarda un canvi. Disposa de mòduls MVC reusables per a estalviar codi.

  El fet de no necessitar de gaires llibreries de tercers ja que ja disposa de mecanismes per a solventar casi qualsevol situació, provoca que el codi base sigui molt més llarg que altres Frameworks

- **Gin:** Mentre que Frameworks com Revel aposten per una solució *all in one* on el Framework és més complex però te moltes opcions per tot, Gin aposta per a solucions més simples i minimalistes aportant les llibreries i funcionalitats indispensables convertint-lo en un Framework lleuger i net.

És simple d'usar i disposa d'una bona documentació per a començar i realitzar tasques més avançades. Es va crear amb un competidor similar al cap que s'explica a continuació, Martini. Gin ha aconseguit ser fins a 40 vegades que Martini i convertir-se en un dels Frameworks més àgils i ràpids.

Tot i aquestes bones característiques, Gin es queda curt en projectes empresarials grans. Necessita de la incorporació de moltes llibreries externes que a vegades poden resultar difícils d'integrar. A més, amb les poques funcionalitats que incorpora, tendeix a sobrecarregar al client amb tasques que generalment fa el servidor.

- **Martini:** De la mateixa manera que Gin, es tracte d'un Framework net, simple i lleuger, però que fa èmfasi en facilitar la integració de llibreries externes per a ampliar funcionalitats convertint-lo en un Framework més similar a Revel.

  Ofereix bona escalabilitat però el seu nivell òptim es situa entre petits grups de programadors i grans empreses, no arribant tampoc al nivell per a projectes grans. Te una bona comunitat i documentació per a trobar respostes a problemàtiques que puguin aparèixer.

  Per temes d'enrutament, Gin usa Httprouter i Martini no, implementant el mateix protocol però amb altres llibreries mes complexes i pesades fent que sigui 40 cops més lent que Gin. No rep manteniment d'actualitzacions des del 2014.

- **Web.go:** Es tracte d'un altre Framework minimalista que incorpora les eines imprescindibles per a la creació d'aplicacions web bàsiques. Per a projectes complexos requereix de la incorporació de llibreries externes. Com a característica rellevant Web.go usa un sistema d'enrutament amb *arbres* provocant una millora d'eficiència, facilitat d'ús, etc.

  Al ser tant simple i lleuger són poques les funcionalitats que aporta de sèrie que no es pugin fer directament amb GO de forma senzilla.

- **Gorilla:** És un dels Frameworks més flexibles a nivell de modularitat. Permet eliminar paquets, afegir extensions, afegir mòduls sense penalitzar el rendiment i millorant la escalabilitat. Es tant modular que s'usa tant per a projectes molt senzills amb petits grups de programadors fins a projectes grans d'empreses.

Una altre de les característiques més rellevants és que suporta *websockets,* un protocol de comunicació que ofereix una connexió *full-duplex* (bidireccional) sobre una única comunicació TCP.
En altres paraules, permet que la comunicació entre les dos entitats (client i servidor) es pugui dur a terme de forma simultània millorant el rendiment.

Com a aspectes més negatius destaquen els aspectes de configuració, que en aquest cas es fan més pesats.

Existeixen altres Frameworks amb característiques similars com Goji, o Beego. El primer és mes similar als més petits, és més aviat un microframework, mentre que Beego es una opció més similar a Revel.

IDES

La majoria dels entorns de desenvolupament compatible amb Golang ja han sigut descrits anteriorment com poden ser: Atom, Eclipse, Sublime Text, Vim o Komodo. Després hi ha altres com:

- **IntelliJ IDEA**: De la companyia ja mencionada Jet Brains. Aquest IDE permet la programació amb Golang però mitjançant la instal·lació d'un plugin.

- **Wide**: És un IDE pensat en el treball en equip per al desenvolupament d'aplicacions web en Golang. Integra Git i altres funcionalitats com debugging, autocompletat, etc.

- **LiteIDE**: Va ser dels primers IDES centrat en Golang. Incorpora sistemes de gestió del codi, línia de comandes, debugger, autocompletat així com suport per a JSON.

- **Visual Studio:** Dins dels nombrosos llenguatges que suporta, Golang és un d'ells. És un dels IDES més complerts que existeixen amb nombroses eines i funcionalitats: debugging, break-points, piles, consola, explorador de fitxers per jerarquies etc.

- **Gogland:** Un altre IDE de la companyia JetBrains explícitament per Golang. Integra totes les eines i funcionalitats dels grans entorns de programació com terminal, debugger, assistència de codi, etc.

A continuació és mostren els punts forts i febles de Golang.

Punts Forts	Punts Febles
• Fàcil d'aprendre. • Molt bones llibreries o *gems*. • Millor que altres llenguatges com Java en *Garbage Collector* o ús de memòria. • Defineix un format de codificació que facilita entendre el codi entre membres d'un equip. • Incorpora un Framework per a Testing a la seva llibreria estàndard.	• Gestió d'errors: Tota funció que pot acabar en error ha de tenir un error com a resultat final. És el sistema usat en comptes de les excepcions. A nivell de rendiment no és un punt dèbil però a nivell de programar alguns programadors ho poden trobar més pesat. • No usa constructors, les variables poden ser usades sense inicialitzar i a vegades deriven en error.

4.2.6 Javascript

És un llenguatge que va sorgir amb el propòsit inicial de programar certs comportaments a les pàgines web arrel d'una interacció de l'usuari. És a dir, va néixer com a llenguatge de *scripting* de la banda de client. Avui en dia és molt diferent, amb la necessitat d'aplicacions web modernes han provocat que Javascript arribi a nivells molt més complexos podent-se comparar amb altres llenguatges de primer nivell.

La visió de Javascript per a programar petits codis per a realitzar accions a les pàgines web s'ha quedat molt curta i avui en dia ja el podem trobar a la banda de servidor, en aplicacions externes a la web com en documents PDF i fins i tot com a llenguatge natiu en sistemes operatius o en grans empreses d'internet com Google, Facebook, etc. que avui en dia, tenen tonelades de Javascript al seu nucli.

Tot i compartir un nom similar amb Java, aquests tenen semàntiques i propòsits diferents. De fet, Javascript es va dissenyar amb una sintaxi similar a la de C. Es compatible amb tots els navegadors actuals on majoritàriament s'executa a diferència dels altres llenguatges com PHP que s'executen a servidor. Més endavant s'explica com també es pot executar a servidor usant Node.js.

Entre altres usos, s 'usa molt per a enviar i rebre informació del servidor amb ajuda d'altres tecnologies com AJAX que també s'explica més endavant. Com a característiques trobem el tipatge dinàmic pel qual una variable pot tenir un tipus diferent de valor en dos moments diferents, és orientat a objectes, permet avaluar expressions en temps d'execució, etc.

Llibreries

Javascript disposa de diverses llibreries molt potents, algunes de les més rellevants s'expliquen a continuació:

- **Cross Browser:** Avui en dia la majora de navegadors son capaços d'interpretar Javascript de forma molt similar, però encara existeixen algunes diferències així com navegadors antics que l'interpreten de forma molt diferent. Per això existeixen un conjunt de llibreries que permeten traduir el codi de cara a la interpretació específica d'un navegador concret.

- **JQuery**: És la llibreria més coneguda de Javascript i existeix com a complement en la majoria de webs que usem a diari. Permet escriure codi compatible i igualment interpretat en tots els navegadors, inclosos els antics. Permet afegir plugins per a la validació de formularis, pas de diapositives, interfícies d'usuari avançades.

- **React:** És una llibreria JS de codi obert dissenyada per crear interfícies d'usuari amb l'objectiu de facilitar la creació d'aplicacions web d'una sola pàgina. És una llibreria gestionada per software lliure. React representa la vista en un patró Model – Vista – Controlador.

- **Polymer:** És una llibreria de codi obert desenvolupada per Google que permet crear elements d'una web sense entrar en un nivell complex.

Frameworks

A continuació es mostren alguns dels Frameworks més usats en Javascript. Estan principalment pensats per al desenvolupament d'aplicacions web d'una sola pàgina com Gmail.

- **Angular:** És un dels més potents i eficients Frameworks de Javascript gestionat per Google que s'usa per a la creació i manteniment d'aplicacions web d'una sola pàgina basades en navegador i patró Model – Vista – Controlador.

Com a principals característiques destaquen la velocitat, rendiment i productivitat. Té funcions de generació de codi en JS a partir de plantilles, executa la vista de l'aplicació en Node.js, .NET i PHP, les aplicacions d'Angular tenen un temps de carga molt baix degut, entre altres coses, al enrutador de components que bàsicament selecciona el fragment de codi que l'usuari necessita per visualitzar i el carrega.

Ofereix una línia de comandes, Angula CLI, que permet desenvolupar ràpidament, afegir components i realitzar tests. Per últim, incorpora les funcionalitats dels IDEs de primer nivell com suggerències de codi, detecció d'errors, etc.

- **Backbone:** És un dels Frameworks més famosos de Javascript. Defensa la idea que totes les funcionalitats del servidor s'han de cridar des d'una API i per això te una interfície RESTful amb JSON. Està dissenyat per a la creació d'aplicacions d'una sola pàgina i per a la sincronització de múltiples clients i servidors. Implementa el patró Model – Vista – Controlador

- **Ember:** És un Framework de codi obert basat en el model Model – Vista – VistaModel que s'explica més endavant en aquest document. Permet als desenvolupadors crear aplicacions web escalables d'una sola pàgina. Ember s'usa en moltes aplicacions famoses com LinkedIn, Vine, Groupon, etc. També permet la creació d'aplicacions d'escriptori o de mòbil. L'exemple més famós per a aplicació d'escriptori és Apple Music.

- **Meteor:** És un Framework per a aplicacions web amb JavaScript de codi obert i usant Node.js el qual es veurà més endavant. Meteor facilita la creació ràpida de prototips i genera codi multiplataforma (Web, Android, iOS). Com a característiques principals s'integra amb MongoDB, usa un protocol (Distributed Data Protocol) per a propagar un canvis a les dades al client sense requerir d'un codi de sincronització, depèn de JQuery al client .

- **Mithril:** És un altre Framework orientat a client per al desenvolupament de aplicacions web d'una sola pàgina amb la característica de ser mes senzill i petit que els altres. És ràpid i incorpora moltes de les funcionalitats de React.

Existeixen altres Frameworks amb característiques similars de construcció d'interfícies d'usuari i aplicacions d'una única pàgina com Vue.js o Aurelia.

A continuació es mostren les tendències d'ús entre els Frameworks de Javascript segons el portal Medium i l'article de Eric Elliott - Top JavaScript Frameworks and Topics to Learn in 2019. A la Figura 47 es mostren les tendències de cerca segons Goolge Trends des de 2014 fins 2019.

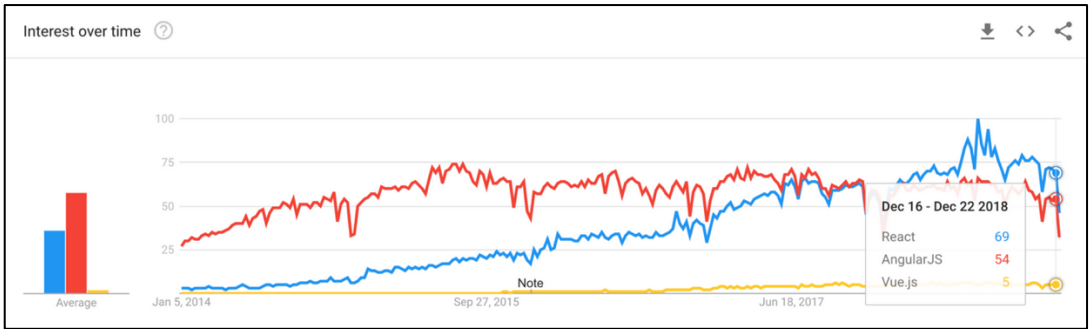


Figura 47 - Portal Medium - Comparació dels Frameworks JS més cercats

En quant a tendències destaquen React i Angular per sobre de la resta. Pel que fa a les descarregues es manté el lideratge d'aquests dos Frameworks. Figura 48.

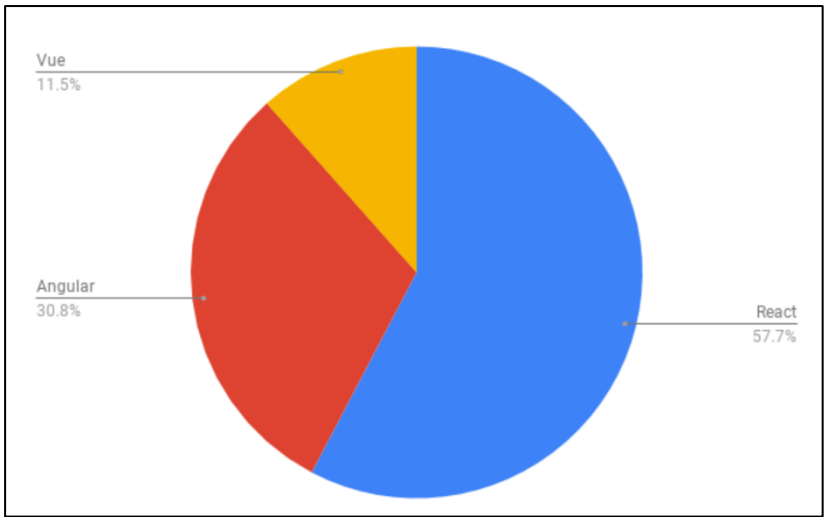


Figura 48 – Portal Medium – Percentatge de descarregues.

Finalment, a la Figura 49 s'observa la tendència pel que fa a ofertes de treball on un altre cop, destaquen React i Angular.

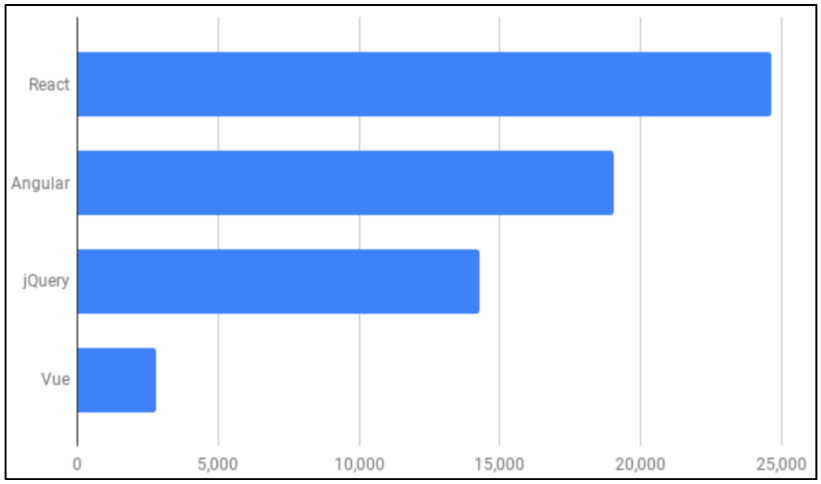


Figura 48 – Portal Medium - Ofertes de treball relacionades

Node.js

Molta gent el descriu com un Framework més de Javascript, però Node.js és un entorn en temps d'execució multiplataforma orientat a servidor i de codi obert. És a dir, permet executar Javascript al *backend,* a la banda del servidor. Per a la interpretació i execució del codi Javascript, Node.js usa el motor v8 de Google, el mateix entorn que usa Google Chrome per a executar Javascript al navegador.

A més, Node.js incorpora molts mòduls que permeten estalviar codi per a la resolució de tasques específiques. Per tant, Node.js no és nomes un entorn d'execució sinó també una llibreria. És ideal per a la creació de Serveis Web basats en REST. També es pot crear un Web Service Soap amb Node.js però ja no es tant senzill i requereix de l'ús de llibreries externes.

A continuació es mostren alguns dels punts fors i febles de JavaScript.

Punts Forts	Punts Febles
• Velocitat. S'executa al navegador del client sense necessitar compilar-se. • Simple. És bastant simple d'aprendre i implementar en comparació amb altres llenguatges. • Es relaciona bé amb altres llenguatges, es pot executar en qualsevol navegador i s'usa per altres tipus d'aplicacions (escriptori, mòbils, etc.) • Comunitat i documentació. Al ser tant usat te una àmplia comunitat i suport. • Redueix la carga de treball del servidor. • Permet crear interfícies avançades per a facilitar l'accessibilitat i l'ús d'una web. • Actualitzacions anuals.	• Al executar-se en el client, es pot usar amb propostes malicioses com instal·lar malware. • Tot i que cada vegada menys, Javascript és interpretat de forma diferent segons el navegador on s'interpreta provocant que una mateixa web es vegi o actuï diferent en dos navegadors diferents.

4.2.7 Elixir

Elixir és un llenguatge funcional (basat en l'ús de funcions matemàtiques) i concurrent que s'executa sobre la màquina virtual BEAM amb llenguatge Erlang. Aquesta màquina virtual, el llenguatge Erlang i Elixir estan pensats per a la concurrència, per al desenvolupament d'aplicacions distribuïdes grans amb tolerància a errors i concurrència de peticions.

Companyies com Pinterest o Discord usen Elixir per les seves aplicacions web. S'ha fet popular aquests últims anys com Go per al desenvolupament de petites aplicacions web per lo simple net i fàcil de mantindré que resulta el codi de Elixir.

Permet la creació de Web Services en Soap amb l'ús de llibreries externes o REST de forma més senzilla. Elixir disposa de diversos Frameworks específics segons el que es vulgui desenvolupar com poden ser:

- Nerves (IoT & Embedded Software)
- Sugar, Plug, Trot (Web)
- Maru, Placid (REST API)

Però entre tots ells destaca un molt per sobre, sens dubte el més famós per el seu ús i potència es Phoenix.

- **Phoneix:** És un Framework per al desenvolupament d'aplicacions web i Web Services. Usa el patró Model – Vista – Controlador i està dissenyat per a poder crear aplicacions d'alt rendiment i escalables. Proporciona comunicació amb clients externs mitjançant websockets o usant polling amb algunes de les funcions de concurrència que disposa el seu llenguatge Elixir.

A continuació es mostren alguns dels punts forts i febles de Elixir.

Punts Forts	Punts Febles
• Poca latència. Processa un gran volum de missatges amb el mínim *delay*. • Codi fàcil d'entendre i és més ràpid que altres llenguatges. • La màquina virtual Erlang que permet gestionar la concurrència i altres funcionalitats • Framework molt potent com Phoenix • Concurrència. Elixir desenvolupa les aplicacions amb els nivells de concurrència més alts. • Tolerància als errors. De forma nativa incorpora mecanismes per a tractar diferents tipus d'errors.	• El codi és fàcil d'entendre però per a programadors d'altres llenguatges la sintaxi pot resultar difícil d'acostumar-se. • Per moltes funcionalitats convé tenir coneixements de Erlang. • Té una comunitat molt més petita que altres llenguatges. • No te tanta demanda.

4.2.8 - Perl

Perl és un llenguatge dissenyat l'any 1987 agafant característiques similars a C i d'altres com Lisp. És un llenguatge imperatiu però amb elements de programació Shell que permeten una sintaxi més flexible. Amb les actualitzacions que va rebent, permet que Perl encara estigui ben situat per al desenvolupament d'aplicacions web modernes on tenen grans interfícies d'usuari i potents *backends.*

Una de les clares característiques rellevants de Perl és el CPAN (*Comprehensive Perl Archive Network*) on es troba la documentació de Perl així com diversos mòduls descargables per a la realització de tasques concretes. Per exemple si es vol desenvolupar un Web Service amb suport JSON el CPAN ens dona opcions instal·lables que poden estalviar moltes línies de codi.

Tot i això, la tendència de Perl respecte anys anteriors segons l'índex TIOBE, un índex de popularitat entre els llenguatges de programació basat en la companyia TIOBE de Eindhoven. Tot i no ser un indicador definitiu si que dona mostres de les tendències entre els llenguatges al llarg dels anys. Igual que les mesures vistes anteriorment, aquest índex recull informació de les cerques de Google, Youtube i molts altres llocs.

L'índex d'aquest mes mostra Perl a la catorzena posició. Dels llenguatges aquí mostrats, només està per sobre de Golang. Si s'analitza la trajectòria al llarg dels anys es veu com aquesta és negativa tal i com es mostra a la figura 49.

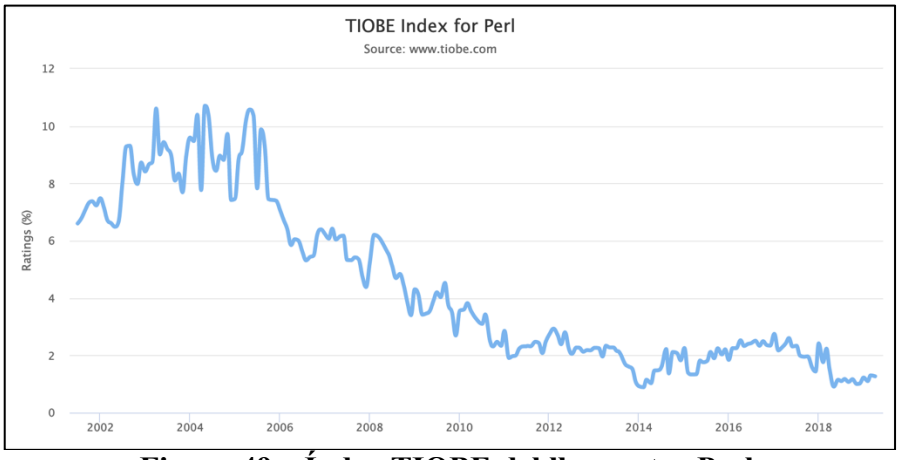


Figura 49 – Índex TIOBE del llenguatge Perl

Any rere any millora incorporant noves funcionalits i apostant per la simplicitat a la hora de programar, disposa de diversos Frameworks i eines per a facilitar el desenvolupament de aplicacions web i Web Services. A continuació es mostren alguns:

- **Catalyst:** És flexible i prou potent com per a desenvolupar des de petites fins a grans aplicacions. Permet afegir extensions i llibreries externes per a expandir funcionalitats.

- **Dancer:** És una opció minimalista tant en sintaxi com en abast, és a dir, està dissenyat per a desenvolupar petites aplicacions web. Tot i això permet afegir plugins per a expandir funcionalitats i que programadors amb experiència puguin construir aplicacions més grans.

- **Mojolicius:** Permet crear aplicacions web en temps real. És de codi obert i permet el desenvolupament de petites i grans aplicacions. Suporta WebSockets HTML5, XML, JSON, CSS3, etc. També disposa de mecanismes natius per a l'ajuda en el desenvolupament d'APIS REST.

A continuació es mostren alguns dels punts forts i febles de perl:

Punts Forts	Punts Febles
• Sembla un llenguatge Shell. • Pels programadors més tradicionals, la seva sintaxi s'assembla més. • És un llenguatge molt potent i versàtil. • És nativament imperatiu però també pot operar com orientat a objectes o funcional .	• Hi ha moltes maneres de fer el mateix. El que pot resultar en mes complexitat per a entendre un codi. • Com a llenguatge d'scripting és mes lent en algunes tasques. • La orientació a objectes no és tant eficient com en altres objectes.

Altres llenguatges poden ser usats per a la creació d'aplicacions web però no son tant recomanables ja que estan principalment pensats per altres propòsits com poden ser c++ . Microsoft ASP.NET és una altre opció viable per a construir una aplicació web però generalment es sempre la menys escollida, era una opció molt més usada a la dècada dels 2000's però amb la consolidació de PHP i Java i l'arribada de Python, Ruby on Rails, etc. és actualment de les opcions menys escollides.

4.2.9 Comparativa entre llenguatges

Vistos els punts forts i dèbils dels llenguatges més usats en desenvolupament web s'intenta, en aquest apartat, mostrar una comparativa final entre ells. Cada un te unes propietats que el poden fer especial en un projecte específic, però a nivell genèric, no hi ha millor manera de comparar-los que amb les pròpies tendències en el mon del desenvolupament web.

El problema de realitzar aquesta comparativa és que es força subjectiva, i canvia amb molta facilitat i freqüència. Existeixen molts portals i índex que la intenten dur a terme i tots obtenen resultats lleugerament diferents. Per tant, el que s'extreu d'aquesta comparativa no és ni molt menys la realitat absoluta sinó una sèrie d'indicadors que junts si que tenen la rellevància com per a observar canvis de tendència.

A continuació es mostren alguns dels indicadors més rellevants. Cal tenir en compte que apareixen molts llenguatges que no formen part del desenvolupament web, o no son els més adequats per a ell com pot ser C.

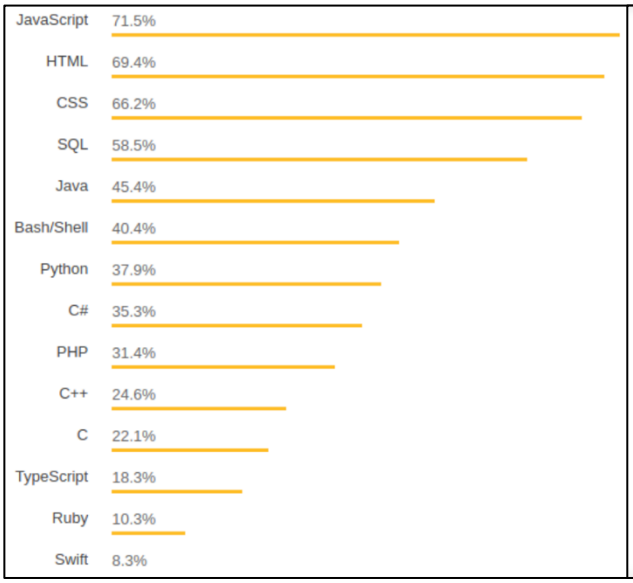


Figura 50 – Portal ValueCoders

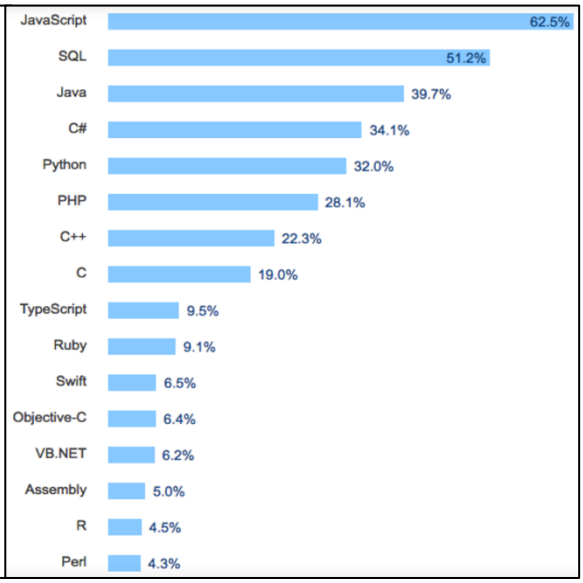


Figura 51 – Portal Medium

Programming Language	Ratings	Change
Java	15.035%	-0.74%
C	14.076%	+0.49%
C++	8.838%	+1.62%
Python	8.166%	+2.36%
Visual Basic .NET	5.795%	+0.85%
C#	3.515%	-1.75%
JavaScript	2.507%	-0.99%
SQL	2.272%	-0.38%
PHP	2.239%	-1.98%
Assembly language	1.710%	+0.05%
Objective-C	1.505%	+0.25%
MATLAB	1.285%	-0.17%
Ruby	1.277%	-0.74%
Perl	1.269%	-0.26%

Figura 52 – Índex TIOBE

4.3 Web Service

Els serveis web son arquitectures orientades a serveis que estableixen relacions de comunicació amb altres sistemes per a l'intercanvi de dades que permeten realitzar determinades funcionalitats. Tenen funcions definides que permeten intercanviar únicament les dades que son necessàries optimitzant el procés. En el seu procés de disseny cal tenir en compte els següents principis entre altres (e-SIRCA-Desarrollo y consumo de Servicios web. Buenas prácticas):

- **Baix acoblament:** És un dels factors crítics. L'objectiu és el de reduir les dependències entre sistemes implicats. Per això, la comunicació s'estableix mitjançant WSDL, un format basat en XML que s'usa per a la descripció de serveis web. Descriu la forma de comunicació, els requisits del protocol i els formats dels missatges necessaris per a interactuar. S'usa sovint amb combinació amb SOAP, el client llegeix el WSDL per a veure quines funcions estan disponibles i usa SOAP per a cridar una d'elles.

- **Reusabilitat:** Els serveis web estan dissenyats pensant en la seva reutilització dins una mateixa aplicació. És a dir, per a que diversos subsistemes accedeixin a la informació per a donar un servei.

- **Interoperabilitat:** Permet que els consumidors i els serveis desenvolupats en tecnologies i plataformes siguin compatibles creant així components totalment independents amb funcionalitat pròpia però que es comuniquen per a crear sistemes més complexos sense preocupar-se sobre com estan construïdes o quin sistema usen els diferents sistemes connectats

- **Granularitat:** Permet dividir les funcionalitats en diferents serveis per a donar una resposta específica amb només les dades necessàries al client.

Existeixen diverses estratègies de disseny a l'hora d'implementar un Web Service:

- **ContactFirst:** Definir les operacions, mètodes i dades necessàries com a fase inicial de disseny abans de passar al codi.

- **ContactLast:** Es desenvolupa una estructura bàsica on s'hi van afegint els mètodes i operacions necessàries.

Estudiant aquests dos mètodes clarament es recomana l'ús del primer, ja que permet aconseguir un millor servei, més robust i més tolerant a canvis i variacions. A més permet disminuir el nombre d'errors i canvis en el desenvolupament.

4.3.1 - Tecnologies de desenvolupament

- **Servidor Rest**: Segueixen la arquitectura REST (Representational State Transfer) i utilitza directament el protocol HTTP per a obtenir dades o indicar la execució d'operacions sobre elles utilitzant formats com XML o JSON. Defineix un conjunt petit d'operacions, les més importants són: POST, GET, PUT i DELETE. S'accedeix a cada recurs mitjançant una URI (Identificador de recurs). Un exemple podria ser: GET /amigos on amigos seria el recurs a obtenir.

- **Soap Server**: *Simple Object Access Protocol* és un protocol estàndard que defineix com dos objectes en diferents processos poden comunicar-se per mitjà de l'intercanvi de dades XML. Està basat en tres parts diferents (Figura 53):

- Sobre: defineix que hi ha al missatge i com processar-lo.
- Conjunt de regles de codificació per a expressar les dades.
- Convenció per a representar la crida a procediments i les respostes.

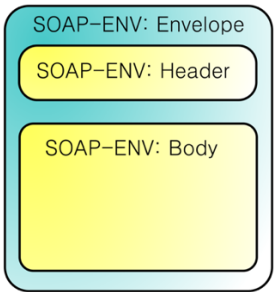


Figura 53 – Estructura en Soap

Vistos els diferents models es mostra a continuació una comparativa entre els dos sistemes:

SOAP	REST
• Es pot consultar la descripció del servei al WSDL. • SOAP incorpora mecanismes per a reenviaments en cas d'errors. • Les dades s'han de representar seguint el protocol de SOAP.	• La seguretat del missatge deriva en el protocol HTTPS punt a punt. En cas d'error s'espera que el client el gestioni. • És més flexible en quant a la representació de les dades. • És més fàcil d'entendre i més simple. • Són més lleugers.

Quan usem un o l'altre?

SOAP

- Si es necessiten mesures extres de seguretat pel tipus de servei implicat, SOAP ofereix estàndards addicionals per garantir el servei.
- En cas de que proveïdor i consumidor tinguin un protocol d'intercanvi de dades SOAP dona especificacions rígides per aquest tipus d'interaccions.
- Si les operacions a executar són més complexes que les CRUD (Create, Read, Update and Delete) SOAP disposa de la estructura per a gestionar-les millors.

REST

- Si es disposa d'una banda ample o recursos limitats. Qualsevol navegador pot interactuar amb un servidor REST ja que usa estàndards GET, PUT, POST i DELETE.
- Per a operacions CRUD que realitzen una tasca concreta de creació, lectura, actualització o esborrat.
- Si les operacions es poden emmagatzemar a caché REST resulta perfecte per les operacions sense estat de REST.

Les màquines que intervenen en la comunicació d'un sistema poden ser molt diferents (Diferents S.O, diferents llenguatges de programació o bases de dades), és per això que per a la comunicació s'usen llenguatges te text pla on destaquen:

- **XML:** Està basat en etiquetes igual que HTML. És més tradicional però també és més avançat en quant a les diverses utilitats per la seva extensió, validació de la informació i sintaxi de les dades.

- **JSON:** És un llenguatge més nou, basat en sintaxi Javascript. Generalment és un més lleuger però requereix de millor càrrega per part del servidor.

4.3.2 Estil de WSDL

Com s'ha explicat anteriorment WSDL permet a un client veure quines funcions hi ha definides al Web Service així com entendre els requeriments i formats de connexió i comunicació amb ell.

En cas d'escollir la tecnologia SOAP per a la implementació del servei web cal tenir en compte quin estil de WSDL triar. El paràmetre binding indica com establir la connexió amb el servidor. Aquest paràmetre es pot configurar amb les següents opcions:

- **RPC/Encoded:**

  El nom de l'operació apareix al missatge per lo que al receptor li és més fàcil enviar el missatge a la implementació de la operació. El WSDL és tant simple com WSDL pot ser-ho. Com a exemple suposem la funció:

  ```
  public void myMethod(int x, float y);
  ```

El RPC/encoded WSDL quedaria:

```
<message name="myMethodRequest">
    <part name="x" type="xsd:int"/>
    <part name="y" type="xsd:float"/>
</message>
<message name="empty"/>

<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
    </operation>
</portType>

<binding .../>
```

Figura 54 – RPC/Encoded WSDL

Al invocar el mètode amb un 5 com a valor del paràmetre X i un 5.0 com a valor del paràmetre y el Soap Message seria el següent:

```
<soap:envelope>
    <soap:body>
        <myMethod>
            <x xsi:type="xsd:int">5</x>
            <y xsi:type="xsd:float">5.0</y>
        </myMethod>
    </soap:body>
</soap:envelope>
```

Figura 55 – Soap Message

Punts febles

- La codificació del tipus (xsi:type = “xsd:int”) que s’observa a la Figura 24 acostuma a ser una sobrecarrega i degrada el rendiment.

- No es pot validar el missatge fàcilment ja que de tot el missatge només les línies <x...>5</x> i <y...>5.0</y> contenen dades definides en un esquema (Schema) la resta del cos (Soap Body) prové de definicions WSDL

- No és compatible amb WS-I, una especificació de Web Services de la indústria Web Services Interoperablity.

- **RPC/Literal**

El resultat del mètode anterior és molt similar:

public void myMethod(int x, float y);

```
<message name="myMethodRequest">
    <part name="x" type="xsd:int"/>
    <part name="y" type="xsd:float"/>
</message>
<message name="empty"/>

<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
    </operation>
</portType>

<binding .../>
```

Figura 56 – RPC/Literal

```
<soap:envelope>
    <soap:body>
        <myMethod>
            <x>5</x>
            <y>5.0</y>
        </myMethod>
    </soap:body>
</soap:envelope>
```

Figura 57 – Soap Envelope

El WSDL segueix sent el més senzill possible i el nom de l'operació segueix apareixent al missatge però en aquest cas, RPC/Literal elimina la informació tipus (xsi: type="x:int") i és WS-I compatible.

Com a punt feble segueix sense poder validar el document fàcilment ja que només els camps <x>5</x> i <y>5.0</y> estan definits dins d'un esquema mentre que la resta del document pertany a definicions WSDL.

- **Document/Encoded**

Es tracta d'un estil no compatible amb la especificació WS-I i que actualment no es fa servir.

- **Document/Literal**

En aquest cas el WSDL generat a partir de la funció anterior varia considerablement:

```
<strong><types>
    <schema>
        <element name="xElement" type="xsd:int"/>
        <element name="yElement" type="xsd:float"/>
    </schema>
</types></strong>

<message name="myMethodRequest">
    <part name="x" <strong>element="xElement"</strong>/>
    <part name="y" <strong>element="yElement"</strong>/>
</message>
<message name="empty"/>

<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
    </operation>
</portType>

<binding .../>
```

Figura 58 – Document/Literal

```
<soap:envelope>
    <soap:body>
        <xElement>5</xElement>
        <yElement>5.0</yElement>
    </soap:body>
</soap:envelope>
```

Figura 59 – Soap Envelope

En aquest cas s'elimina la codificació tipus. Es pot validar amb un validador XML i és compatible WS-I.

Punts febles

- El WSDL es complica, però al no ser llegit per humans es tracta d'un punt feble poc important.

- El nom de la operació ja no apareix per lo que el enviament es pot complicar.

4.3.4 Connexió amb la base de dades

El Web Service serà l'encarregat de dur a terme la connexió amb la base de dades per a la inserció i recuperació de les dades.

Segons el llenguatge escollit i la base de dades amb la que es treballi (MySQL, MongoDB, etc.) es podrà fer amb funcions ja incorporades o caldrà incorporar llibreries externes. Un cop realitzada la connexió amb la base de dades es definiran les funcions d'inserció, actualització i obtenció de dades. El client cridarà a la funció corresponent amb els paràmetres necessaris, el servidor executarà la funció i retornarà, si escau, els valors obtinguts.

Segons si es tracta d'un servidor REST o SOAP les dades es retornaran en formats diferents i al client haurà de tractar el resultat per a obtindré els valors finals.

4.4 Aplicació Web

En aquest apartat s'estudiaran diverses opcions per a la construcció d'un portal web. Es tindrà en compte les restriccions descrites en aquest document, és a dir, no serà un portal purament informatiu sinó que haurà de complir una sèrie de funcionalitats. A més consumirà informació d'un Web Service.

4.4.1 – Patrons de disseny

Model – Vista – Controlador (MVC)

El model vista controlador és el patró de disseny d'arquitectura web més usat i implementat pels Frameworks descrits anteriorment. Permet separar l'aplicació en 3 capes (Figura 60):

- **Model**: És la capa que representa la informació amb la que opera el sistema, és a dir, la base de dades. Gestiona tots els accessos a la informació tant per a recuperar-la com per a inserir-la. Rep les instruccions del controlador i envia la informació resultant a la vista corresponent.

- **Controlador:** És la capa que gestiona tot el sistema. Reacciona a events provocats per l'usuari (Clic sobre un enllaç, enviar un formulari, etc.) i els relaciona amb una funció que pot interactuar amb una nova vista o amb el model. El controlador forma part del que s'anomena *Backend* i pot estar programat amb diferents llenguatges descrits anteriorment (PHP, Java, Python, etc.)

- **Vista:** És la capa visual amb la que interacciona l'usuari final**.** Rep la informació a mostrar del controlador, model o dels dos. La vista forma part de l'anomenat *Frontend* i està programada en HTML, Javascript i CSS. Les vistes també suporten inserció de codi Java, PHP, Python i altres per a determinades funcionalitats amb el tractament de les dades de forma transparent a l'usuari final.

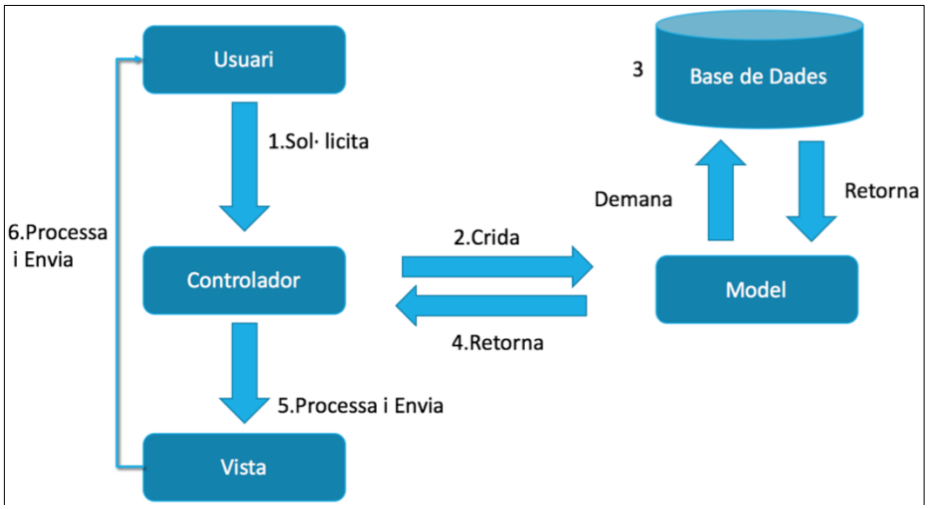


Figura 60 – Exemple Model Vista Controlador

Aquest patró s'usa tant degut a que és molt senzill d'entendre i permet estructurar el sistema. Tot i això, a vegades no representa amb prou fidelitat on s'han d'ubicar certes funcions. És el programador el que decideix per exemple si vol ubicar el tractament de les dades obtingudes de la base de dades al model o al controlador. Per aquest motiu i molts altres existeixen variants o altres models diferents al model-vista-controlador.

A la Figura 61 s'observa una altre aproximació on el model envia directament les dades a la vista.

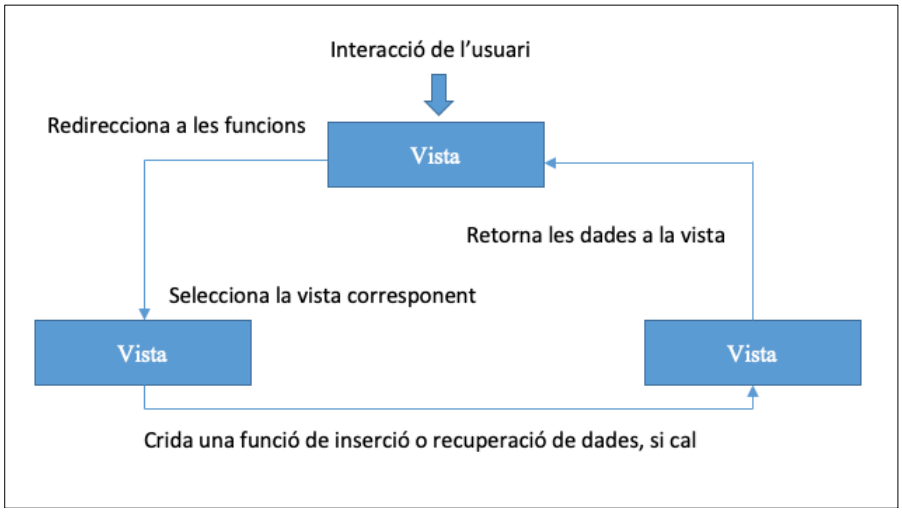


Figura 61 – Model – Vista – Controlador

Model – Vista – Adaptador (MVA)

És una variant del MVC que separa el sistema en 3 capes però una d'elles, l'adaptador, s'encarrega de gestionar totes les peticions de l'usuari evitant així, que la vista i el model interactuïn directament.

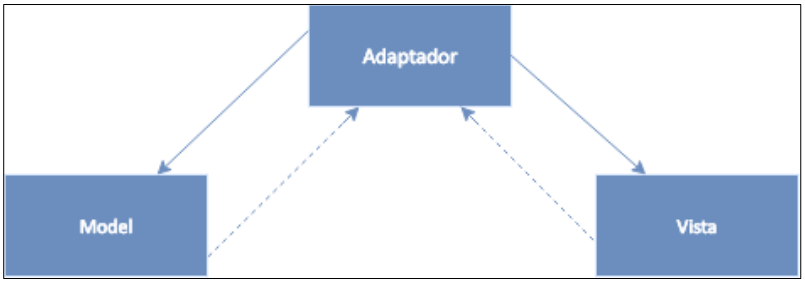


Figura 62 – Esquema Simplificat MVA

Model – Vista – Presentador (MVP)

És una altre variant del MVC creada per Taligent (Companyia subsidiària de IBM) per a C++, C# i Java que de nou, separa el sistema en 3 capes amb els objectius de maximitzar la quantitat de codi verificable de forma automatitzada i separar la lògica de negoci de la interfície d'usuari per a entendre i mantenir el codi de forma més simple.

```
public class DomainView : IDomainView
{
    private IDomainPresenter domainPresenter = null;

    ///<summary>Constructor</summary>
    public DomainView()
    {
        domainPresenter = new ConcreteDomainPresenter(this);
    }
}
```

Figura 63 – Instanciació del presentador en C#

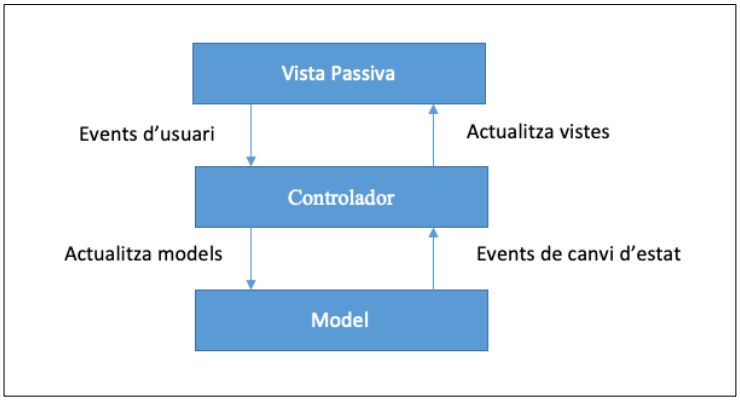


Figura 64 – Esquema Model – Vista – Presentador

Model – Vista – VistaModel

Una altre variant del MVC on existeixen les mateixes 3 capes de separació però en aquest patró de disseny, quan hi ha un canvi a la vista sobre una dada, aquesta s'actualitza directament al model. El controlador passa a ser un "*ViewModel*" tal i com s'observa a la Figura 65.

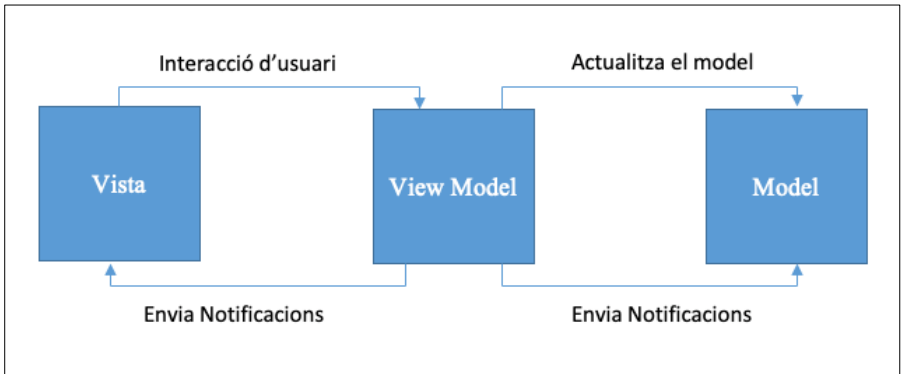


Figura 65 – MVVM

Vistos alguns dels patrons de disseny més usats, s'expliquen a continuació les tecnologies per a la construcció de les pàgines web.

4.4.2 - Sistemes de Gestió de Contingut (CMS)

Quan es pensa en portals web estàtics (Informatius) i de petites funcionalitats (E-commerce) la tecnologia que s'usa més avui en dia són els CMS (Content Management System).

Fins fa relativament poc, construir una web estàtica requeria coneixements d'alt nivell de HTML, Javascript i CSS per la part visual i de PHP, Java o altres per a codificar webs amb certes funcionalitats i tractament de dades. A més, el fet d'actualitzar les tecnologies emprades en el desenvolupament de la web també requerien de molta feina i temps.

A mesura que les webs creixien es feia més indispensable una eina que per ella sola permetés en primer lloc la creació de la web sense la necessitat de dominar les tecnologies descrites anteriorment, la seva gestió, actualització i administració. Aquestes eines són els CMS.

Existeixen CMS dissenyats per a tasques més específiques com gestions de correu o tendes online però tots satisfan tres funcionalitats bàsiques:

Creació de la web

Permet la construcció d'un portal web amb grans aspectes moderns de disseny amb un coneixement molt bàsic dels llenguatges normalment necessaris (HTML, Javascript, CSS, PHP, etc.). Com a característica principal destaca que els CMS separen el disseny visual del contingut, és a dir, es podrà anar variant el disseny sense que afecti al contingut sinó que aquest s'adaptarà al nou disseny.

Dit això, tot i que cada vegada existeixen temes (aparences visuals) més personalitzables, aquestes ens deixen un conjunt de elements i disseny "tancat" que no podrem personalitzar al mateix detall que una web programada a codi.

Per a poder personalitzar i afegir el contingut s'usa el gestor de continguts el qual ens deixa escollir les pàgines a modificar i afegir o modificar el seu contingut. La majoria de continguts es poden afegir mitjançant *Drag&Drop* i modificant la seva configuració com s'observa a la Figura 66.

Els CMS permeten expandir les funcionalitats de la plantilla mitjançant l'ús de Plugins. Els Plugins són fragments de codi desenvolupats per tercers que s'afegeixen als arxius del tema escollit per a afegir funcionalitats.

Existeixen Plugins de pagament i gratuïts. Abans d'instal·lar-ne un, caldrà contrastar les opinions i funcionalitats que aporta ja que pot fer malbé la resta de la web.

Un altre cop, els plugins poden solventar necessitats de funcionalitats però es pot donar el cas de que el que necessitem per a la web que es vol desenvolupar no es pugui adquirir per mitjà de cap Plugin.

En cas de voler sortir dels límits de la plantilla per a poder introduir elements o funcionalitats no disponibles s'hauran de modificar els arxius que de configuració del Tema o dels seus Plugins. Per fer-ho són necessaris alts coneixements de llenguatges com PHP i les modificacions acostumen a ser més difícils que programar la funcionalitat fora d'un CMS.

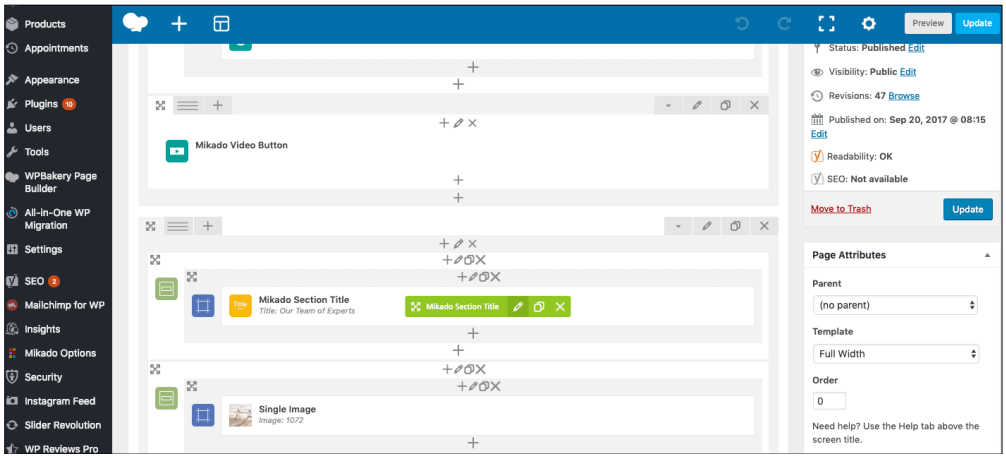


Figura 66 – Exemple gestió de contingut

Gestió i Manteniment de la web

Els CMS permeten mantenir tots els components de la web de forma ràpida i senzilla. El propi CMS ens avisa de quan un recurs te una actualització disponible i serà la nostre decisió si instal·lar-la o no. Tots els components instal·lats a la nostre web estan desenvolupats per tercers que s'encarreguen de mantenir-los al dia.

Abans d'actualitzar un tema o Plugin caldrà, però, estar segurs que es compatible i pot funcionar correctament amb la resta del nostre entorn instal·lat. A la Figura 67 s'observa un exemple, en aquest cas de Wordpress, del panell de control i dels avisos sobre actualitzacions disponibles.

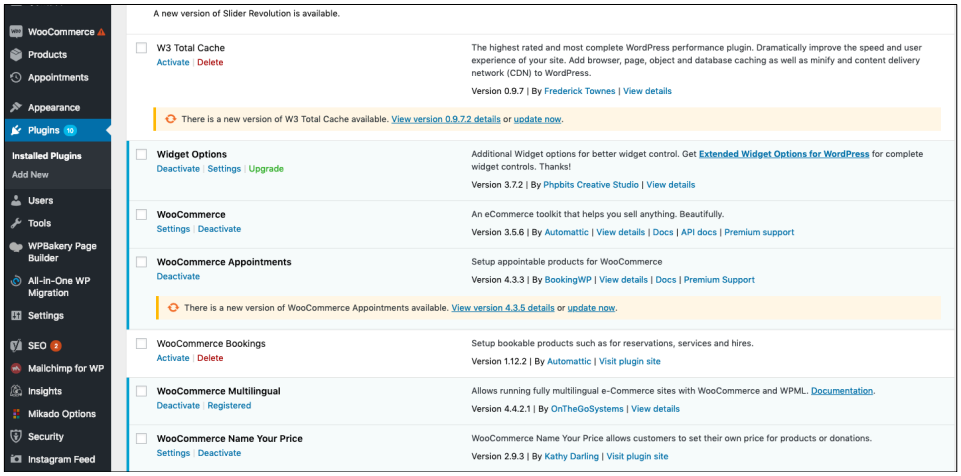


Figura 67 – Exemple panell de control Wordpress

Administració

El CMS ens permet una administració total de tots els elements que formen la nostra web. Des de la creació de nous perfils per a que puguin accedir al panell de configuració i realitzar tasques a instal·lar eines de control i monitorització de la pàgina (Posicionament SEO, Google Anlaytics, etc) així com la modificació d'arxius de configuració i contingut directament a codi. (Figura 68)

Figura 68 – Exemple administració d'arxius del tema

Vistes les característiques, sembla que amb un CMS es pot construir i gestionar qualsevols tipus de web. Es cert, però tenint en compte que en casos especials caldrà afegir i modificar codi per a crear les funcionalitats i requeriments que els plugins i temes no ens poden solucionar.

Donat aquest cas, ja es feina del programador o encarregat del projecte decidir si usar un CMS per a tenir un 50-70% de la feina "feta" mitjançant plantilles i plugins i modificar el percentatge resultant a codi o al veure que no es podrà complir tots els requeriments amb el CMS implementar el sistema a codi.

La primera opció obliga a conèixer i usar les llibreries i funcions, així com el llenguatge, del CMS emprat, mentre que la segona, dona llibertat a usar el llenguatge i Framework que més s'ajusti a les nostres necessitats.

Exemples de CMS

Com hem vist els CMS compleixen les mateixes característiques generals, per tant, no hi ha una opció millor que altre més enllà dels gustos personals. Tot i això, si que existeixen CMS que incorporen de fàbrica funcionalitats específiques sense requerir la instal·lació de plugins de tercers ni modificacions de codi.

- **Wordpress:** És tracte del CMS més usat degut, entre altres, a la seva senzillesa a l'hora de usar-lo. És ideal per als usuaris que no tinguin coneixements de programació web. Consta de un ampli catàleg de plugins i plantilles de pagament que donen la possibilitat de fer-ho pràcticament tot així com una àmplia comunitat d'usuaris. Idealment pensat per a portals web petits però amb capacitat i eines per a fer "qualsevol" cosa.

- **Joomla:** Aquest CMS requereix d'una corba d'aprenentatge més gran degut a la versatilitat dels seus components per a canviar els aspectes visuals de disseny.

- **Drupal:** És un dels CMS més complexos que hi amb una corba d'aprenentatge més amplia degut a la seva capacitat per a construir llocs web més grans i complexos. La principal característica és la integració de components i eines per a l'optimització del posicionament web (SEO).

- **Prestashop:** És un CMS pensat per a comerços electrònics incorporant de fàbrica eines i components per a facilitar la seva creació sense la necessitat d'instal·lar plugins externs.

- **Moodle:** És un CMS pensat per a la creació i gestió de cursos online. Incorpora de fàbrica eines, recursos i components per a la gestió d'usuaris (alumnes, usuaris, professors), gestió de cursos i altres funcions sense la necessitat d'instal·lar plugins externs.

Existeixen altres CMS com Shopify, Blogger, Magento i altres amb característiques específiques com creació de e-coomerce, blogs, etc. Una característica dels CMS aquí descrits és que estan tots programats amb PHP.

Tot i que els CMS més famosos estiguin escrits en PHP no vol dir que no hi hagin d'altres en altres idiomes. A continuació es mostren alguns dels CMS més famosos en altres llenguatges.

- **Python**: Wagtail (Django), Mezzanine (Django), MoinMoin, Plone
- **Pearl**: Sellerdeck (ecommerce), Bricolage, blosxom, WebGUI, SPINE
- **Java**: Magnolia, Xwiki, OpenKm, OpenWGA, Hippo CMS, Alfresco
- **Javascript:** Ghost, Wiki.js, TiddlyWiki
- **Ruby:** Alchemy, Radiant, Publify, Refinery
- **ASP.NET:** DNN, Umbraco, BetterCMS; mojoPortal, Orchard Project

4.4.3 - Pàgines web personalitzades a codi

Un cop vistos els CMS i les seves característiques s'analitzen ara diverses opcions per a la creació de pàgines web a codi. Primer de tot, en una pàgina web feta a codi és molt més important la etapa de disseny i estructuració de tots els fitxers. Caldrà dissenyar una base de dades amb la que s'interactuarà per a la inserció i obtenció de dades. En el cas descrit anteriorment d'un sistema més complex on la web consumeix un web Service, aquesta base de dades serà manipulada per aquest últim.

Un cop definides quines dades necessitarem i el seu format, cal escollir el llenguatge i l'entorn de programació (IDE). Cal diferenciar entre el Front-end i el Back-end els quals ja s'han descrit anteriorment en aquest document. Per al primer, no hi han tantes opcions a escollir ja que està bastant estandarditzat :HTML5, CSS3 i Javascript seran els llenguatges necessaris per a poder dissenyar l'aspecte visual i d'interacció amb l'usuari de la nostre web.

Pel que fa al back-end, com ja s'ha vist anteriorment existeixen moltes més variants que cal analitzar però que a "grosso-modo" no tenen millores entre si sinó que depèn del programador i del projecte en si.

Per a entendre millor el que un Framework ens pot proporcionar ja no només amb llibreries sinó amb la estructura de fitxers s'analitza amb més detall un exemple amb PHP i Codeigniter :

PHP - Codeigniter

Com s'ha explicat anteriorment, és un el llenguatge més usat per els CMS així com per una gran varietat de pàgines i serveis web. És fàcil d'usar i aprendre, ràpid i compatible amb la gran majoria de servidors. Requereix d'un servidor amb Apache o IIS amb llibreries PHP, és compatible amb amplis formats de bases de dades i la orientació a objectes.

Al instal·lar Codeigniter sens crea automàticament una estructura de fitxers com la que s'observa a la Figura 69. Els fitxers es troben dividits principalment en arxius de configuració, arxius d'aplicació (Controladors, Vistes, etc.) i *assets* (On trobem les imatges externes, vídeos i fitxers d'estils i comportament CSS i Javascript. Recordem que Codeigniter esta dissenyat seguint el patró Model – Vista – Controlador.

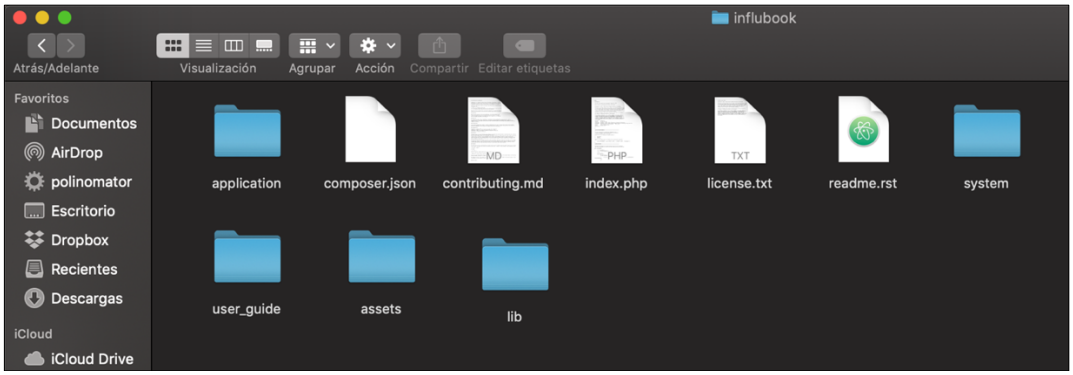


Figura 69 - Estructura de fitxers Codeigniter

Una de les funcionalitats comentades destacades és la d'afegir llibreries externes que ens dones eines i funcions per a funcionalitats específiques. A la Figura 70 s'observa com es poden carregar aquestes llibreries, en aquest cas la de *session* que ens permet crear. mantenir i destruir una sessió d'usuari.

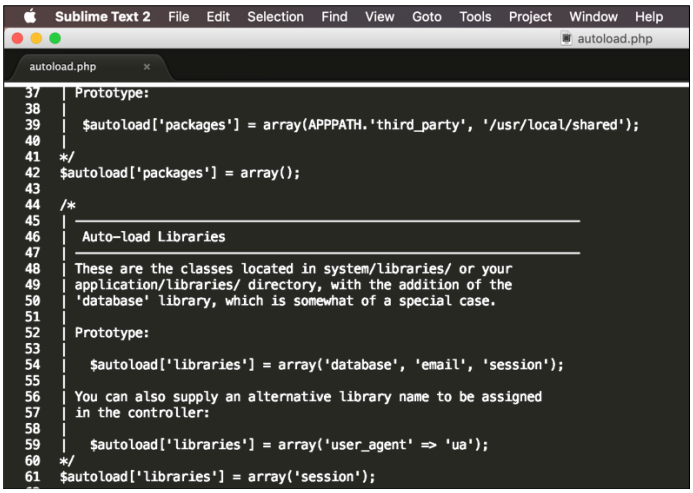


Figura 70 – Autoload de llibreries externes

Al entrar la url de la nostre web el primer que carrega és el fitxer de rutes que indica on trobar el controlador principal dins els directoris application/controllers/. Al controlador definirem les funcions amb el nom exacte de la url, per exemple, suposem que el nostre controlador s'anomena ES.php i que volem definir una pàgina de contacte, al controlador definirem una funció contactar que cridarà a la vista encarregada de mostrar el HTML de la pàgina de contacte. La vista es troba a application/views/ i Codeigniter ja s'encarrega de fer l'enrutament per no haver de posar la ruta sencera.

Per tant, la url quedaria així: www.lamevaweb.com/es/contactar on 'es' fa referència al controlador i contactar a la funció descrita dins el controlador. A la figura 71 es mostra un exemple de controlador.

```
public function guia(){
    $data['tituloHeader']='Jazzac';
    $data['description']='Guia del Jazz de Cataluña';
    $data['pos'] = 'somos';

    $data['images'] = array(
        array('1', 'grupo'),
        array('2', 'grupo'),
        array('3', 'grupo'),
        array('4', 'festival')
    );


    $this->load->view('include/ESheader_inicio', $data);
    $this->load->view('ESinicio', $data);
    $this->load->view('include/ESfooter', $data);
    $this->load->view('include/footer.php', $data);
}
```

Figura 71 – Exemple controlador

4.4.3.1 - .htacces

El fitxer .htacces és un fitxer ocult que conté una sèrie de directrius per al servidor web. S'usa per a restringir l'accés a carpetes, personalitzar les URL, les pàgines d'error, etc.

A la Figura 72 es mostra un exemple de fitxer .htacces en aquest cas del CMS Wordpress.

.htaccess

```
                        Require not ip 115.159.147.138
                </RequireAll>
        </IfModule>
        <IfModule !mod_authz_core.c>
                Order allow,deny
                Allow from all
                Deny from env=DenyAccess
                Deny from 120.27.100.100
                Deny from 115.159.147.138
        </IfModule>
# END iThemes Security - Do not modify or remove this line

RewriteEngine on
RewriteCond %{HTTPS} off [OR]
RewriteCond %{HTTP_HOST} !^www\. [NC]
RewriteRule ^ https://www.1850urbanspabarcelona.com%{REQUEST_URI} [R=301,L,NE]


# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>

# END WordPress
```

Figura 72 – Exemple .htaccess

4.4.3.2 - Bootstrap

Bootstrap aporta un conjunt d'eines HTML, CSS i Javascript que permeten dividir el contingut de la pàgina en columnes d'una certa amplada. Aquestes columnes canvien de posició a mesura que la mida de la pantalla varia. En altres paraules, permet de forma molt senzilla crear pàgines *responsive design,* adaptables a la mida de la pantalla.

Conté plantilles de pàgines senceres, formularis, botons, menús de navegació i molts més elements. És el segon projecte més destacat de Git-hub, de codi obert i es usat per la NASA com a exemple destacable.

Prové d'una sèries de fulles d'estil on es defineixen unes classes o columnes que tenen un ample determinat i un posició concreta a la pantalla. Per exemple la classe col-sm-7 és una columna on es poden disposar elements i ocuparà de la meitat de la pantalla al marge dret.

Un full d'estil anomenada boostrap.less té les configuracions generals d'ample de la pàgina i les diferents columnes. Modificant aquest arxiu podem modificar l'ample de les columnes així com decidir a partir de quin ample de pantalla volem que es recol·loquin els elements.

Quan la mida de la pantalla canvia, les columnes se situen en una nova posició, adaptant-se unes després d'altres mantenint un disseny organitzat. D'aquesta manera quan l'usuari accedeix amb una pantalla petita no cal que faci zoom per centrar la pantalla i veure be els elements, sinó que aquests es disposen automàticament.

Tal i com s'explica al capítol 1, avui en dia l'accés a internet des de mòbils i tabletes és molt ampli i cada cop major, així que és de especial importància desenvolupar un bon *responsive-design* .

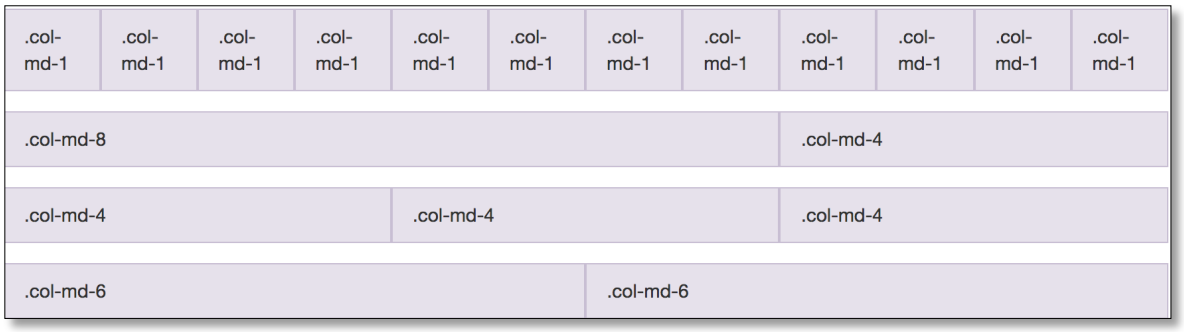


Figura 73 – Exemple diferents columnes Bootstrap

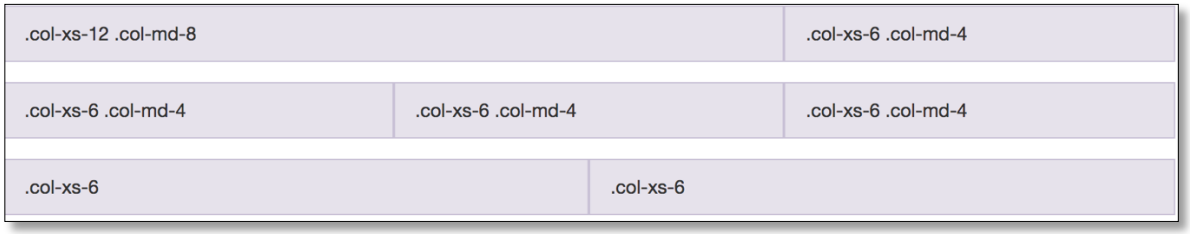


Figura 74 – Exemple diferents columnes Bootstrap

Els exemples de les figures anteriors 73 i 74 són només alguns exemples de tots els tipus de columnes que existeixen. Es poden observar les diferents nomenclatures:

- **xs:** Extra Small Devices. Per a dispositius amb pantalles més petites que 768px com els mòbils.

- **sm:** Small Devices. Per a dispositius amb pantalles compreses entre 768px i 992px com les Tablets.

- **md:** Medium Devices. Per a dispositius amb pantalles compreses entre 992px i 1200px com els portàtils i ordinadors de sobretaula petits.

- **lg:** Large Devices. Per a dispositius amb pantalles més grans que 1200px com ordinadors de sobretaula amb pantalles grans.

4.4.3.3 - CSS (Cascading Style Sheets)

És un llenguatge de disseny gràfic per a crear la presentació de pàgines web. És a dir, es combina amb HTML per a crear l'aspecte visual (Colors, Posicions, Mides, etc.). Està present a pràcticament la totalitat de les webs. El CSS es pot incorporar directament sobre el fitxer HTML però generalment, al necessitar diverses classes s'acostuma a incorporar en un fitxer apart i incorporar-lo des del fitxer HMTL.

El funcionament és el següent, al HTML es crea l'element a mostrar, per exemple, un menú. Sobre aquest element s'afegeix el terme *class="myMenu"* el que indica que existeix una classe CSS que marca l'estil d'aquell element. Al fitxer CSS s'afegeix la classe amb un '.' Davant del nom seguit dels paràmetres a configurar. També es pot fer mitjançant un id, en aquets cas, al fitxer CSS cridarem l'element amb el símbol '#'. (Figura 75)

```
.footerList{background-color:#6f2828; font-size: 18px; color:white; bottom:0;min-height:
.footerText{font-family:'Open Sans',sans-serif; color:white; font-size:13px;}
.square{background-color:#D8D8D8; border-color: black; border-radius: 50px; width:700px;
.squareCompra{background-color:#D8D8D8; border-color: black; border-radius: 50px; width:
.square2 {background-color:#E3E2E2; border-radius:5px; border-color: black; opacity:0.8;
.square2 p{margin-left: 30px; margin-top: 0; margin-bottom: 0;}
.square3 {background-color:#E3E2E2; border-radius:5px; border-color: black; opacity:0.8;
.square3 p{margin-left: 30px; margin-top: 0; margin-bottom: 0;}
input[type="submit"]{ margin-left:5px; background:#BD6D61; border:0; border-radius:2px;
input[type="submit"]:hover{background:#BA5F51;}
.btn-normal:hover{background:#086A87 !important; }
@import url(http://fonts.googleapis.com/css?family=Montserrat+Alternates);
.loginForm { display:inline; margin-right: 0px;}
.navbar #secondary-nav2 { width: 140px; display: inline-block; margin: 25px 0 0;}
.navbar #secondary-nav {width:240px; display:inline-block;margin:25px 0 0}
.navbar #secondary-nav li{margin-right:3px; float:right;}
.navbar #secondary-nav li:last-child{margin-right:0}
.navbar #secondary-nav li a{font-weight:600;font-size:13px}
.navbar #secondary-nav li a:hover{text-decoration:underline}
```

Figura 75 – Exemple fitxer CSS

4.4.3.4 - HTML (HyperText Markup Language)

És un llenguatge de marques per a la creació de pàgines web. Defineix la estructura bàsica d'espais i elements que conformen la vista d'una web. A partir d'allà, com hem vist s'afegeixen estils amb CSS i interaccions amb Javascript.

HTML intenta mantenir-se simple incorporant tota la resta de continguts de llibreries i arxius externs. És a dir, suporta la incrustació de codi d'altres llenguatges com CSS, JavaScript o PHP però s'intenta afegir el mínim possible de forma directe i importar el necessari de fitxers i llibreries externes.

A la Figura 76 es mostra un exemple d'un fitxer HTML.

```
<!DOCTYPE html>
<html lang="es">
<body>
    <form action="<?php echo base_url('es/busqueda'); ?>" method="post">
        <div class="field" id="searchform" style="width:90%;">
          <input type="text" style="width:90%;" name="busqueda" id="searchterm" placeholder="Buscar" />
          <button type="submit" id="search"><i class="fa fa-search" aria-hidden="true"></i></button>
        </div>
    </form>
        </br>

        <div class="row  animated fadeInUp">
            <div class="container">
                <div class="square2">
                    </br>
                    <center><h2>Consulta Específica</h2></center>

                    <div id="menu">
                        <ul>
                            <span><li class="nivel1"><a href="#" class="nivel1">Artistas</a>
                            <!--[if lte IE 6]><a href="#" class="nivel1ie">Opción 1<table class="falsa"><tr><td><![endif]-->
                                <ul>
                                    <span><li><a href="<?php echo base_url("es/consultas/musicos"); ?>">Músicos</a></li></span>
                                    <span><li><a href="<?php echo base_url("es/consultas/grupos"); ?>">Grupos</a></li></span>
                                    <span><li><a href="<?php echo base_url("es/consultas/management"); ?>">Management</a></li></span>
                                    <span><li><a href="<?php echo base_url("es/consultas/productores"); ?>">Productores</a></li></span>
                                </ul>
```

Figura 76 - Exemple de fitxer HTML

4.4.3.5 - AJAX (Asynchronous JavaScript And XML)

És un conjunt de tècniques de desenvolupament web per a crear aplicacions interactives. Aquestes aplicacions s'executen al client, és a dir, al navegador i de forma asíncrona manté una connexió amb el servidor en segon pla el que permet realitzar canvis sobre les pàgines sense necessitat de recarregar-les millorant la interactivitat, velocitat i usabilitat de les aplicacions

Que sigui asíncron vol dir que les dades es demanen al servidor en segon pla però la pàgina segueix responent sense interrompre. Es pot configurar de forma síncrona el que implica que la pàgina queda inactiva fins que rep la resposta del servidor.

Normalment les crides a AJAX es fan des de Javascript que ja ha sigut explicat anteriorment.

4.4.3.6 – Servidor Local

MAMP

En les fases de disseny i implementació es probable que encara no es disposi d'un domini i servidor on allotjar la pàgina web. Donat el cas, es pot configurar una plataforma de probes amb algun proveïdor de serveis de forma gratuïta o es pot començar el desenvolupament en local.

En aquest últim cas es pot crear un servidor local usant Xampp en sistemes operatius Windows o MAMP per a sistemes Mac. Aquest software usa *Apache*[11] i l'adreça de localhost 127.0.0.1 per a configurar el servidor. Per a una correcte configuració cal seguir els passos següents on es posa com a exemple Mac OS X.

- Editar el fitxer /etc/hosts afegint l'adreça de localhost i el nom que farà referència a aquesta adreça. A la Figura 77 es pot observar un exemple on apareixen diverses pàgines totes referenciades a la localhost.

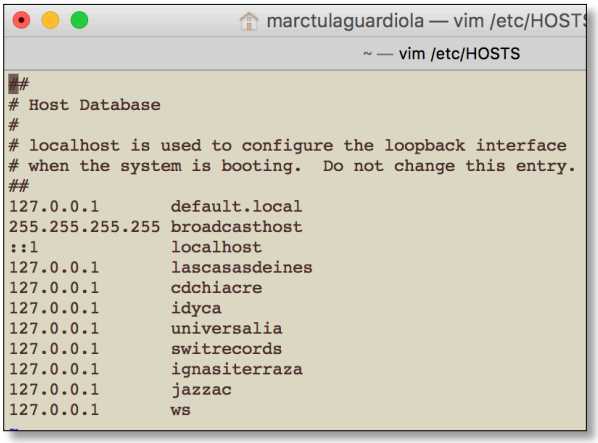


Figura 77 – Fitxer /etc/Hosts

- Accedir a */Applications/MAMP/conf/apache/httpd.conf* i descomentar la línia: Include'*/Applications/MAMP/conf/apache/extra/httpd-vhosts.conf*'

- Obrir el fitxer */Applications/MAMP/conf/apache/httpd.conf* i afegir la configuració que es veu a la Figura 78.

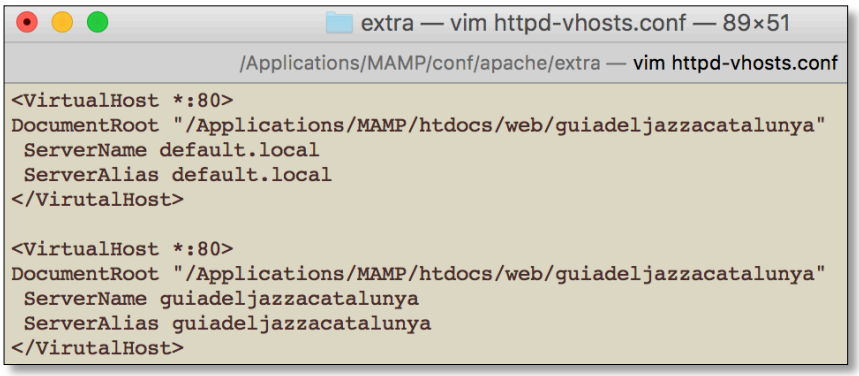


Figura 78 – Configuració d'un VirtualHost

4.4.3.7 - FTP

Un cop es tingui contractat un allotjament web els canvis produïts al nostre servidor local els podrem pujar al servidor mitjançant una connexió FTP (File Transfer Protocol) on es farà una transferència dels arxius modificats.

Aquesta tasca es pot fer de forma manual accedint mitjançant el navegador o amb programes especialitzats com FileZilla que ofereixen funcionalitats de Drag&Drop que faciliten la cerca i transferència de fitxers. (Figura 79)

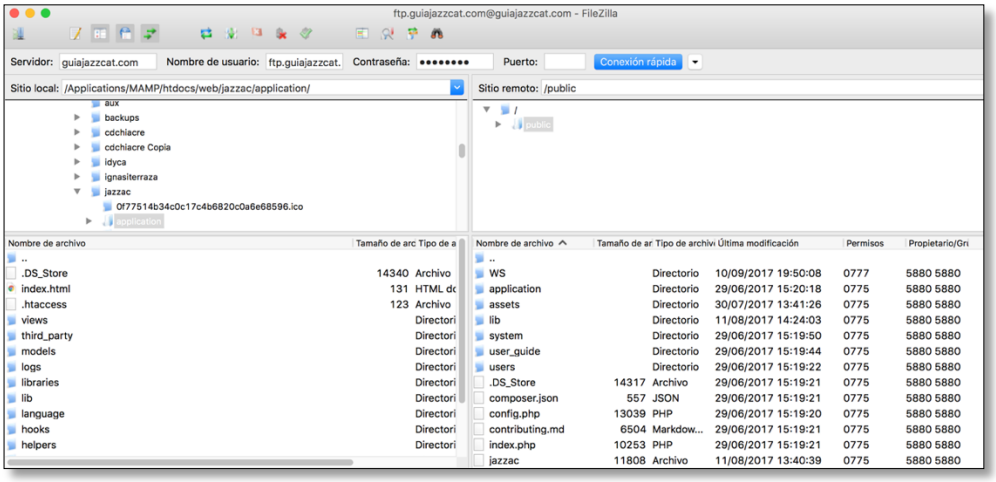


Figura 79 – Captura Filezilla [9]

4.4.3.8 - SEO (Search Engine Optimization)

El SEO fa referència a la posició on apareix una pàgina web en un motor de cerca com pot ser Google i depèn d'un conjunt d'accions i regles al construir la pàgina. Avui en dia, tot negoci amb presència online, aplicacions web, etc. tenen un o més competidors, per tant, es indispensable facilitar l'accés als usuaris a la informació o servei que dona la nostra web per davant de la dels competidors. És per això que el SEO és un dels aspectes més demanats en el desenvolupament web i existeixen empreses que només es dediquen a ell.

El posicionament atorgat directament de la manera en que està construïda la web s'anomena posicionament natural o orgànic i depenen de la indexació que fan les *aranyes (*Aplicacions que recorren totes les webs) per als motors de cerca.

Existeixen diverses tècniques per a millorar el SEO orgànic el millor possible. Aquí es descriuen unes quantes:

- *Responsive Design* on el contingut s'adapta a la mida de pantalla. Es penalitza les webs que no s'adaptin a dispositius mòbils.

- Paraules claus: Cal incloure al *head* dels fitxers html les paraules clau que resumeixen el contingut de la web: *<meta keywords:"Surf, Platja...">* i a cada pàgina una descripció de les funcionalitats i objectiu de la pàgina incloent el màxim nombre de paraules clau. Ex: *<meta description="El surf a Barcelona...">*

- Fitxer *robots.txt*: És un fitxer que s'inclou al directori *root* (principal) amb totes i cada una de les diferents *url* que conformen la pàgina per a que les aranyes dels motors de cerca les trobin i es pugin indexar.

- Títols: es recomana posar títols (Tags: h1, h2, h3 i h4) a cada pàgina amb les paraules clau corresponents.

- Domini i url incloent paraules clau: també és important que les paraules clau apareguin al domini de la pàgina així com evitar que tingui moltes X.

- Velocitat: És important que la pàgina respongui de forma fluida carregant el contingut de forma ràpida. Per aconseguir velocitat caldrà prendre mesures com imatges optimitzades en mida i pes, comprimir els arxius per a reduir el pes de la pàgina o usar una CDN que acosta les dades a l'usuari final.

- No ha d'haver contingut ocult de cara a l'usuari.

- Seguretat: Si la pàgina disposa d'un certificat de seguretat SSL serà millor considerada.

- Realitzar actualitzacions de contingut.

- LinkBuilding: es tracta d'una tècnica que consisteix en aconseguir que altres pàgines enllacin amb la nostra. Quant més important sigui la pàgina que enllaça la nostra més valor tindrà (Figura 80).

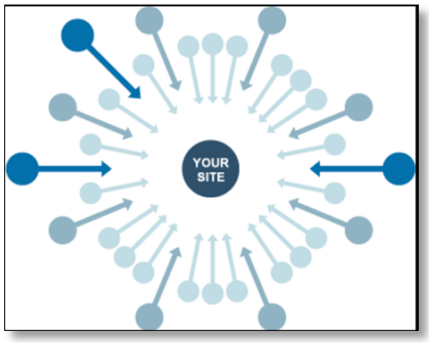


Figura 80 – Exemple Link Building

4.4.3.9 - SEM (Search Engine Marketing)

El SEM fa referència al posicionament web però tenint en compte aspectes externs i no interns de la pàgina. És a dir, mesura la notorietat que te la pàgina a la xarxa. Per a millorar-lo cal tenir enllaços que referenciïn la nostre web com a la tècnica anterior linkbuilding.

- Presència a les xarxes socials on la gent pugui visitar la web, comentar, compartir, etc.
- Aconseguir referències de pàgines importants relacionades amb el sector.
- Blog de contingut variat amb temàtica relacionada i paraules clau rellevants.
- Publicitat amb anuncis com Google Ads, publicitat a instagram i altres xarxes socials, etc.

4. 5 Aplicació Mòbil

En aquest apartat s'estudiaran diverses opcions per a la construcció d'una aplicació mòbil. Es tindrà en compte les restriccions descrites en aquest document, és a dir, no serà un portal purament informatiu sinó que haurà de complir una sèrie de funcionalitats. A més consumirà informació d'un Web Service.

Abans de començar amb la implementació és altament recomanable definir l'abast i funcionalitats de l'aplicació. Per a representar el funcionament i abast de l'aplicació es comença dissenyant el diagrama de classes.

Un diagrama de classes UML (Unified Modeling Language) és un tipus de diagrama que representa la estructura del sistema mostrant les seves classes, atributs, mètodes i les relacions entre ells.

Ens permet tenir una representació de la mida i funcionalitats que tindrà l'aplicació i permet estalviar errors de producció així com desenvolupar més ràpid. Per a la realització d'aquests diagrames el Software més recomanat és StarUML disponible per a Windows i Mac OS X. A la Figura 81 s'observa un exemple d'ús amb aquest software.

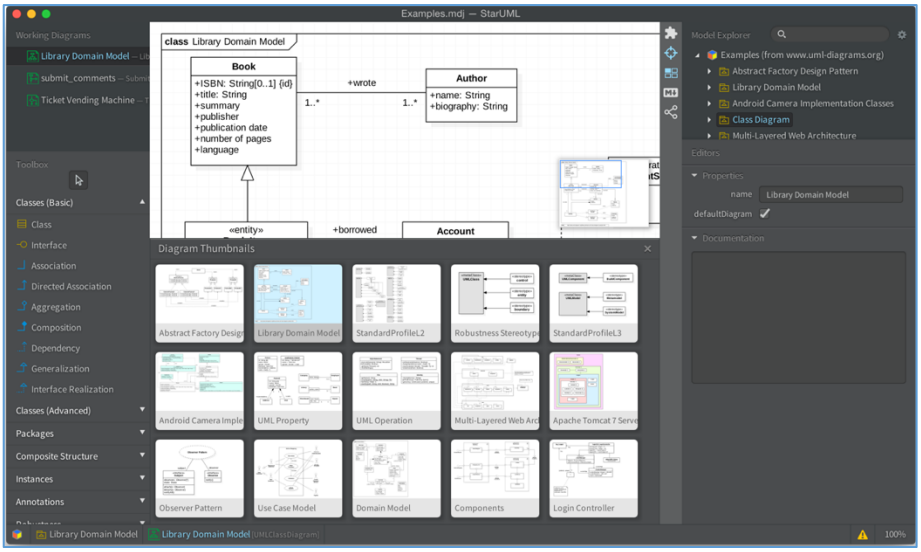


Figura 81 – Exemple Star UML

4.5.1 Android

Android és un sistema operatiu mòbil desenvolupat per Google amb Kernel Linux i de codi obert. És un sistema molt més personalitzable que el seu competidor iOS el qual s'explica més endavant, avui en dia, Android està present a un gran nombre de terminals diferents representant la majoria de la quota de Smartphones.

Les eines necessàries per a la construcció de l'aplicació venen integrades al SDK (Software Development Kit) amb un depurador de codi, diverses llibreries específiques, un simulador d'un Smartphone, etc. Android Studio és el IDE per defecte que integra totes aquestes funcionalitats i moltes més. A la Figura 82 es mostra un exemple amb Android Studio.

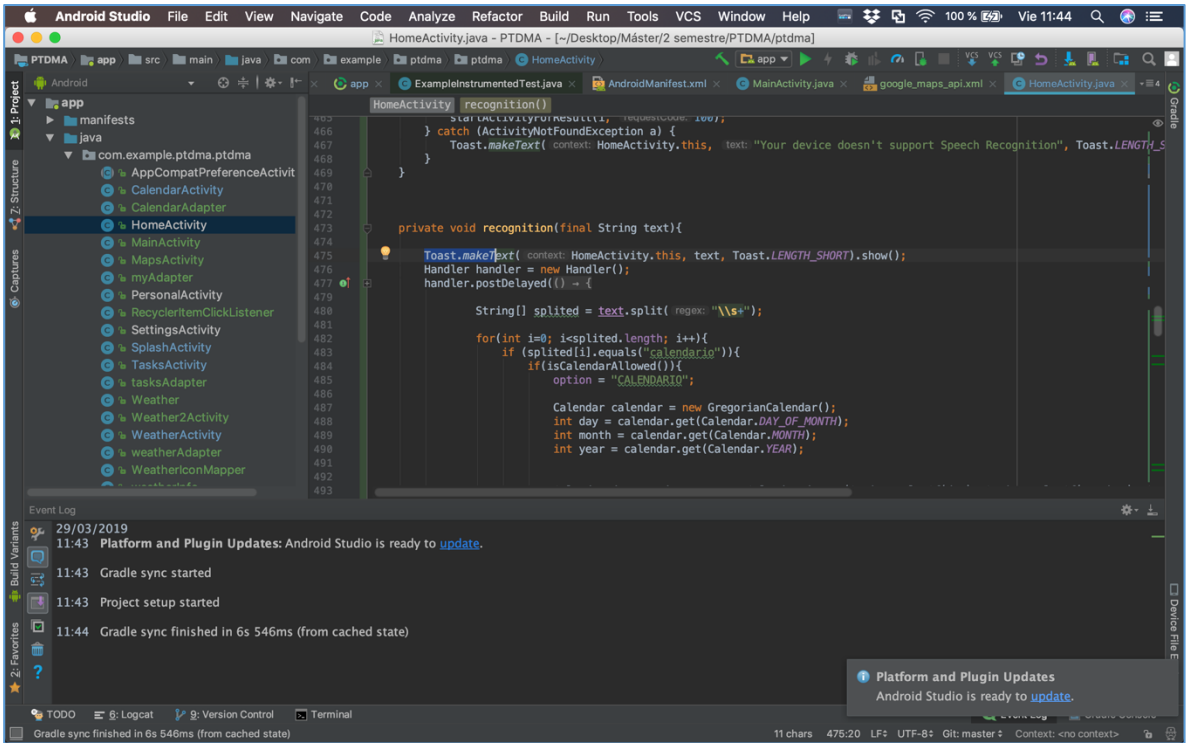


Figura 82 – Exemple android studio

Es poden usar també altres entorns de desenvolupament amb extensions com Eclipse amb la extensió ADT.

Al programar nativament en Android apareixen una sèries d'avantatges i desavantatges que es mostren a continuació.

Avantatges

- S'aconsegueix una interfície millor acabada.
- Es disposa de diversos controls i eines per a l'ajuda del desenvolupament.
- Podem accedir a recursos locals de forma més segura i ràpida (GPS, Camera, etc.)
- Es pot desenvolupar en qualsevol màquina que suporti Java (Linux, Mac OS X, Windows) en canvi iOS requereix d'una mac per a programar nativament en Objective-C.
- No hi ha una barreja de llenguatges i eines diferents.

Desavantatges

- No tenim un accés universal. És a dir només podran instal·lar la aplicació els smartphones amb una SDK d'Android igual o superior a la usada per el desenvolupament.
- Les actualitzacions de l'aplicació depenen de l'usuari final.

Com a eines principals de configuració destaquen Android SDK Manager que permet escollir el *target* de sistema on es vol executar l'aplicació. Es pot instal·lar tantes com es vulgui i cada una pot venir amb noves funcionalitats. AVD Manager permet crear emuladors amb el sistema SDK mida de pantalla i altres aspectes configurables. Amb el emulador podem observar el comportament i la disposició dels elements com si fos un entorn real.

Per altre banda la instal·lació d'una aplicació Android és molt senzilla i si es disposa d'un Smartphone amb una SDK acceptada per l'aplicació es pot connectar via USB i automàticament Android Studio el detecta, instal·la la APK (Fitxer executable) i executa l'aplicació.

Al crear un projecte amb Android estudio aquest queda estructurat en un sistema de fitxers on trobem arxius de configuració, fitxers java (Activities), recursos de l'aplicació com imatges, separats en diferents directoris.

Les activities són els fitxers que codifiquen el comportament de les vistes amb una interacció de l'usuari. Les aplicacions es poden veure com una sèrie d'activities. Les vistes es creen amb fitxers xml usant elements predefinits.

A la Figura 83 es mostra el cicle de vida d'una activitat.

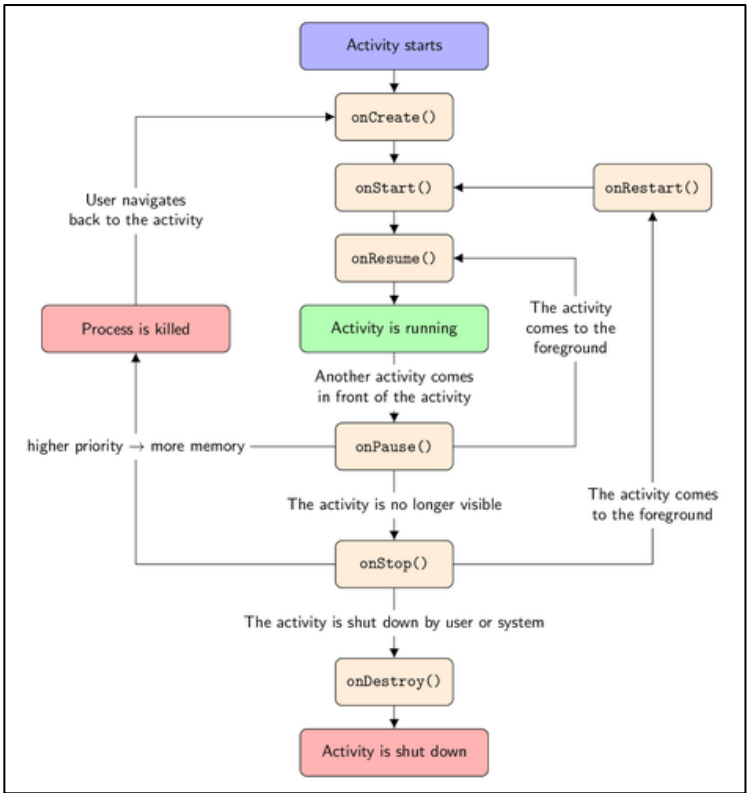


Figura 83 Cicle de vida d'una activitat

Google Play

És la plataforma de distribució digital d'aplicacions mòbils per als dispositius (Smartphones i Tablets) amb sistema operatiu Android. Permet descarregar les aplicacions gratuïtes o de pagament.

El procés per a publicar una aplicació a la plataforma es força senzill comparat amb el seu principal competidor Apple Store. El desenvolupador necessita un compte de desenvolupador de Google, seguidament cal pagar la tassa de 25 dòlars que Google Play tarifa als desenvolupadors. Un cop abonada la tassa caldrà seguir els passos que permeten introduir títol, descripció, països on es vol distribuir, preu de venta (On un 30% de la recaudació se la queda Google) i pujar la APK.

4.5.2 iOS

iOS és el sistema operatiu creat per Apple desenvolupat originalment per al iPhone i actualment també a iPads i iTouch. Ocupa un alt percentatge del mercat de Smartphones i Tablets degut a la gran qualitat d'aquest productes, els únics on es pot instal·lar aquest sistema.

Per al desenvolupament d'aplicacions natives per aquest sistema existeixen, principalment, dues opcions:

4.5.2.1 Objective-C

És un llenguatge orientat a objectes que utilitza el Framework *cocoa*[26] per a utilitzar una sèrie de funcions ja definides que faciliten molt la feina a l'hora de la codificació. A continuació s'observen els diferents tipus de fitxers i algunes propietats d'aquest llenguatge.

Principalment hi ha dues classes de fitxers, els *headers* (.h) els quals defineixen una classe amb les seves funcions i propietats, i els *implementation files* (.m) els quals implementen les funcions definides al .h (Veure Figura 84 i Figura 85).

Propietats: les propietats són la informació que conté l'objecte instanciat. La classe de l'objecte es troba definida al Framework. Es declaren amb l'etiqueta @property seguits del tipus (NSString, NSInteger, etcètera.)

Imports: per a vincular classes i poder cridar funcions d'altres arxius s'usa la instrucció #import seguit del nom del fitxer. El nom s'escriu entre cometes (" ") per a indicar que el fitxer es troba al projecte, o entre operador de més gran i més petit (<>) per a importar fitxers de sistema (del Framework).

```
8
9  #import <UIKit/UIKit.h>
10
11 @interface ViewController : UIViewController
12     @property (nonatomic, weak) IBOutlet UILabel *titol;
13     @property (nonatomic, weak) IBOutlet UITextField *mail;
14     @property (nonatomic, weak) IBOutlet UITextField *password;
15     @property (nonatomic, weak) IBOutlet UIButton *loginButton;
16     @property (nonatomic, weak) IBOutlet UIButton *nonRegistered;
17     -(IBAction) signInButtonTouched;
18     -(IBAction)openDaleDietrichDotCom:(id)sender;
19
20 @end
```

Figura 84 – Exemple fitxer .h

```
#import "ViewController.h"
#import "manager.h"
#import "mainTableViewController.h"
#import <QuartzCore/QuartzCore.h>

@interface ViewController ()

@end

@implementation ViewController

#pragma mark - configs

- (void)viewDidLoad {
    [super viewDidLoad];
    UIColor *colour = [[UIColor alloc]initWithRed:111.0/255.0 green:40.0/255.0 blue:40.0/255.0 alpha:1.0]
        ;
    self.view.backgroundColor = colour;
    self.titol.textColor = [UIColor whiteColor];


    UIColor *colour2 = [[UIColor alloc]initWithRed:189.0/255.0 green:109.0/255.0 blue:97.0/255.0 alpha:
        1.0];
    self.loginButton.backgroundColor = colour2;
    self.nonRegistered.tintColor = [UIColor whiteColor];
    // Do any additional setup after loading the view, typically from a nib.
    //self.loginButton.layer.corenrRadius = 2.0f;
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

Figura 85 – Exemple fitxer .m

Elements de navegació: a continuació s'expliquen alguns dels elements més utilitzats per al desenvolupament de la aplicació.

ViewController: presenta una vista bàsica, una pantalla on disposar elements per on es vulgui.

TableViewController: presenta una vista dividida en cel·les configurables on es poden disposar botons i altres elements. És ideal per a la creació de llistats.

Navigation Bar: la Navigation Bar és una barra de navegació que apareix al marge superior de les vistes. Serveix per posar els botons d'anar enrere, títol i altres elements.

Segues: les segues permeten el salt d'una pantalla a una altre, existeixen diverses maneres per a mostrar una nova pantalla:

- Push: la nova pantalla apareix de dreta a esquerra.
- Modal: la nova pantalla apareix d'abaix adalt
- Popover: Mostra una finestra emergent connectada a la principal.
- Custom: aquesta opció permet crear una animació per a l'aparició de la nova pantalla.

4.5.2.2 - Swift

És un llenguatge multiparadigme creat per Apple enfocat al desenvolupament d'aplicacions per a iOS i macOS. Està dissenyar per a integrar-se amb el Framework Cocoa i Cocoa Touch. Pot usar qualsevol llibreria d'Objective-C i cridar funcions en C.

Swift apareix amb una aparença molt similar a Objective-C en quant a core, però intentant solucionar problemes d'aquest. Amb la seva arribada els desenvolupadors d'iOS van començar ràpid el debat de quin llenguatge era millor.

Tot i l'excitament d'una nova tecnologia que permet construir aplicacions de forma mes senzilla s'era escèptic amb si realment era una tecnologia que arribava per quedarse o acabaria desapareixent. Amb el pas del temps la resposta és clara, en només tres anys Swift va experimentar el creixement més ràpid vist mai en un llenguatge de programació.

Tot i això, Apple segueix donant suport a Objective-c per als programadors veterans d'aquest llenguatge i perquè segueix sent una opció molt vàlida per al desenvolupament d'aplicacions iOS.

A la hora d'escollir quin llenguatge usar, des de la perspectiva empresarial es te en compte el cost de desenvolupament, la seva duració, i les opcions d'innovació que presenta el llenguatge. Per altre banda des de la perspectiva dels desenvolupadors es té en compte la simplicitat de la sintaxi, un compilador que ajudi amb els errors, que sigui segur, etc.

A continuació es mostra una comparativa de la empresa Altexsoft del 2018 on es comparen els dos llenguatges en diversos aspectes, per a poder triar quin convé més segons el context del projecte. Altexsoft és una consultora de tecnologia que opera des del 2007. (Figura 86)

Figura 86 - Comparativa entre llenguatges de la empresa altexsoft Juny 2018

Com s'observa a la Figura 86 són molts els aspectes on Swift treu avantatge sobre Objective – C on destaquen la menor complexitat, el major rendiment, la seguretat i la facilitat de manteniment.

Swift és pot programar des de entorns de desenvolupament disponibles per altres plataformes que no siguin mac com Atom o AppCode (de la companyia JetBrains) mentre que objective-c requereix de Xcode. Tot i això, és altament recomanable usar Xcode com a entorn de desenvolupament d'una aplicació iOS.

Com a Android Studio, Xcode disposa d'una sèrie d'eines per a facilitar el desenvolupament de l'aplicació. Per al desenvolupament de les interfícies gràfiques disposa d'un sistema de *Drag&Drop* on podem anar arrastrant elements i disposar-los com vulguem. També permet la simulació de la aplicació en emuladors d'iPhone, iPad, etc.

A la Figura 87 es mostra un exemple d'entorn amb Xcode, en aquest cas amb Objective-C

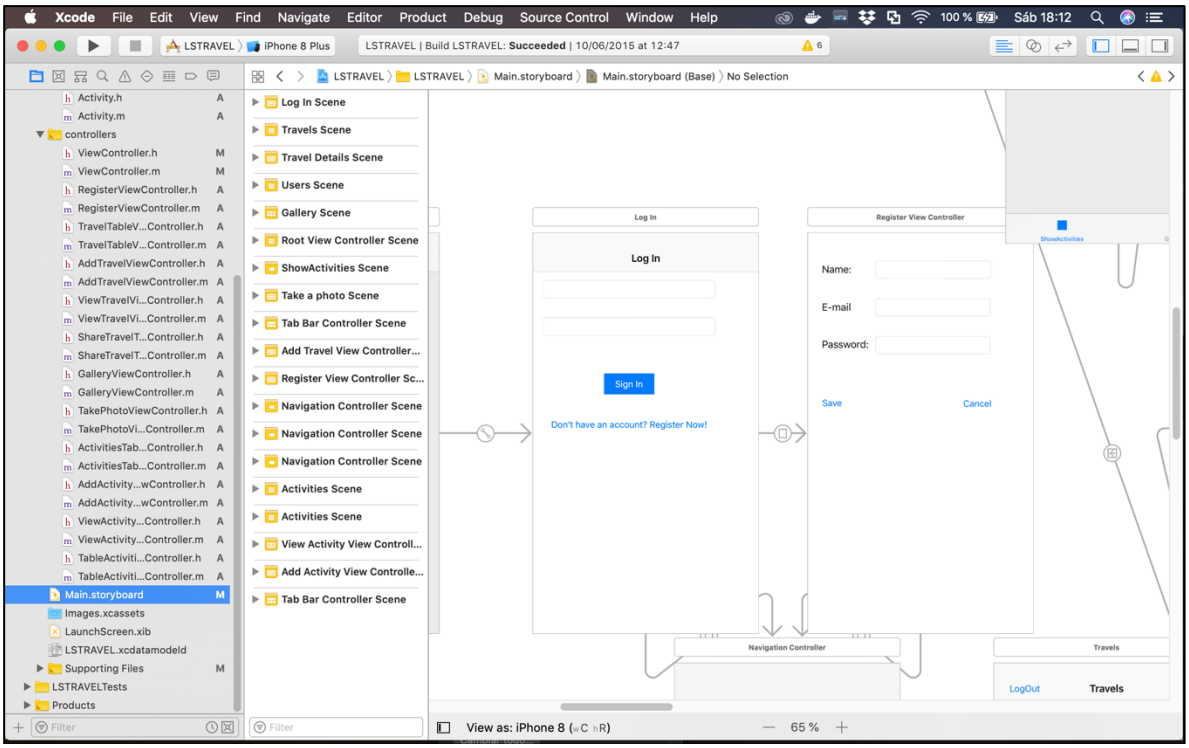


Figura 87 – Storyboard XCode

Apple Store

Apple Store és el servei competidor del descrit anteriorment Google Play però per a les aplicacions d'iOS. El 70% del benefici que recauda una aplicació se'l queda el venedor de l'aplicació mentre que el 30% restant és per Apple.

Pel que fa a les polítiques d'acceptació d'aplicacions són molt més restrictives que a Google Play. És necessari presentar una aplicació que segueixi estrictament les guies d'estil d'Apple així com estar registrat com a desenvolupador. La tarifa bàsica per a poder penjar una aplicació és de 99$

4.5.3 - Aplicacions no natives

Existeix la possibilitat de crear aplicacions híbrides que combinen característiques de les aplicacions web i les aplicacions natives per a mòbils. Principalment combinen els llenguatges HTML, Javascript i CSS de les aplicacions mòbils així com la seva característica multiplataforma i l'accés a elements del dispositiu mòbil de les apps natives, és a dir, no s'executen des del navegador.

La principal característica negativa és que amb aquestes aplicacions no és pot arribar a un aspecte visual i funcional tant professional com de forma nativa. Més aviat estan pensades per a petites aplicacions que no tenen funcionalitats complexes i que interessa que es puguin executar des de qualsevol plataforma.

Alguns exemples d'eines per a la programació d'aplicacions no natives són PhoneGap, JQuery Mobile, appery.io, iBuild App i com a característiques comparteixen l'ús de llenguatges com Javascript, HTML5 i CSS, facilitats com *drag&drop,* etc.

Base de dades

Per a guardar les dades necessàries per al correcte funcionament de l'aplicació web i mòbil cal utilitzar una base de dades connectada al Webservice. La pàgina web i l'aplicació cridaran una funció del Webservice i aquest executarà una funció per a la inserció o consulta de dades.

Les bases de dades són un conjunt de dades emmagatzemades sistemàticament per al seu posterior ús. Utilitzen Sistemes Gestors de Bases de Dades com *MySQL* els quals són un conjunt de programes que permeten l'emmagatzematge, modificació i extracció de la informació d'una base de dades.

La informació s'emmagatzema en estructures anomenades taules, les quals estan formades per columnes amb el nom del atribut i el tipus (INT, VARCHAR, etcètera.). Per a accedir a les taules i recuperar informació s'utilitzen *querys* en llenguatge *SQL*. A continuació es poden veure les funcions més utilitzades en les *querys.*

4.6 Bases de Dades

4.6.1 Model conceptual de base de dades

El model de la base de dades és on queda definida l'estructura de taules on es guarda tota la informació. Per a relacionar taules existeixen les diferents dependències i claus:

Claus

- **PK (Primary Key):** l'atribut únic que no es pot repetir al llarg de la taula. Normalment s'assigna un 'id', però també es pot usar un atribut que es sàpiga segur que és únic i no es pot repetir. Per exemple en un ciutadà podria ser el DNI.

- **PK/FK (Primary Key/Foreign Key)**: per a relacionar taules, cal que la clau primària d'una estigui referenciada a l'altre. A continuació s'expliquen les diferents relacions i com es relacionen les claus a cada una.

Relacions

Herència: a la relació d'herència existeix una taula 'pare' que té uns atributs comuns amb les taules 'fills'. Els 'fills' hereten els atributs i poden tenir alguns específics per a ells. Un exemple seria la classe assignatura amb atributs comuns com el nom, el nombre d'alumnes etcètera. I existirien els fills matemàtiques, programació les quals heretarien els atributs d'assignatura i podrien tenir alguns de propis específics només per a ells i que no són compartits pels altres (Figura 88).

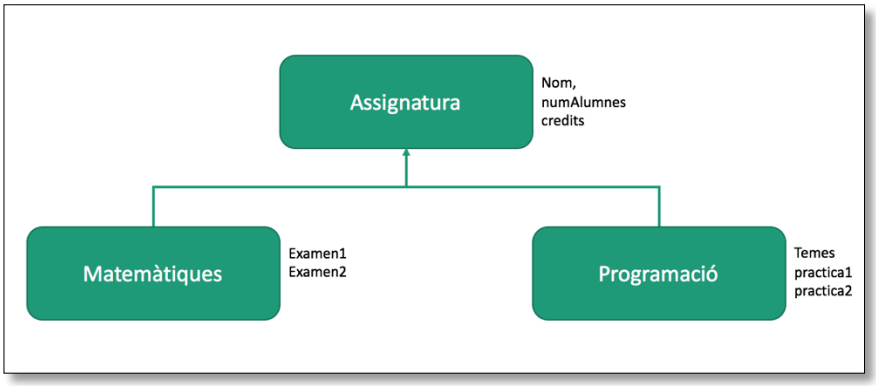


Figura 88 – Relació d'Herència entre taules

1:1: a la relació 1:1 s'especifica que una entitat tindrà relació amb només una altre entitat. Per exemple, un cotxe te un únic motor. En aquest cas es pot posar la PK de qualsevol de les taules com a PK/FK a l'altra (Figura 89).

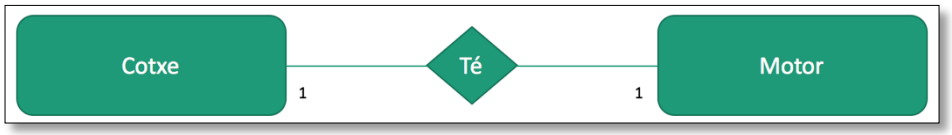


Figura 89 – Exemple relació 1:1

1:N: A la relació 1:N s'especifica que una entitat tindrà relació amb N entitats d'una altre. Per exemple, un alumne estarà matriculat a varies assignatures. En aquest cas la PK de l'alumne s'ha de replicar a Assignatura com a PK/FK (Figura 90).

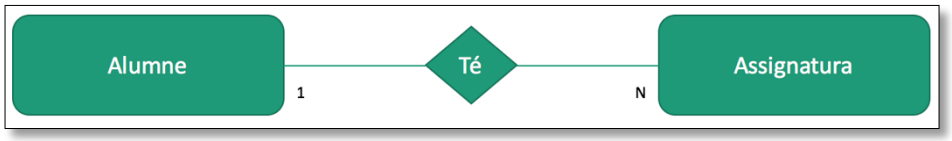


Figura 90 – Exemple relació 1:N

N:M: a la relació N:M s'especifica que diverses entitats d'un tipus estan relacionades amb varies entitats d'un altre tipus. En aquest cas, de la mateixa manera que en la relació 1:1 la PK de qualsevol de les taules relacionades ha d'estar especificada a l'altre com a PK/FK.

Querys

SELECT: operació per a seleccionar atributs de la base de dades. Cal acompanyar la instrucció amb el nom dels atributs a seleccionar i el nom de la taula on es troben. Ex: SELECT nom, cognom FROM usuari.

INSERT: operació per a la inserció d'atributs a la base de dades. Cal acompanyar la instrucció amb el valor dels atributs a inserir i el nom de la taula de destinació. Ex: INSERT into usuari VALUES Marc, Tula.

DELETE: operació per a esborrar elements d'una taula.

WHERE: operació per a indicar alguna característica dels atributs a seleccionar, esborrar. Ex: Select nom FROM usuari WHERE edat=25.

4.6.2 Model Relacional

El model relacional és un grau entremig entre el model conceptual de molt alt nivell i completament abstracte i el model físic de molt baix nivell. En ell queda reflectit la relació entre taules tot veient la seva estructura interna, ja no es veuen els rombes de relació sinó que aquestes queden patents al veure les vinculacions entre claus (PK, PK/FK) a les diferents taules.

4.6.3 Model Físic

El model físic és directament la implementació de la base de dades en aquest cas en MySQL. Queden definides totes les taules i procediments. Tal i com es veu a continuació, existeixen eines que generen automàticament el codi MySQL facilitant la feina de creació de taules i vinculació entre elles.

4.6.4 Eines de creació de Bases de Dades

Tal com s'ha explicat anteriorment, existeixen diverses eines que faciliten la creació, gestió i modificació de bases de dades. Normalment disposen d'un entorn gràfic amb diverses opcions per a crear taules, inserir dades, importar i exportar fitxers, testejar *querys* i moltes altres opcions. Alguns exemples són MySQL Workbench, veure Figura 91, o PHPMyAdmin

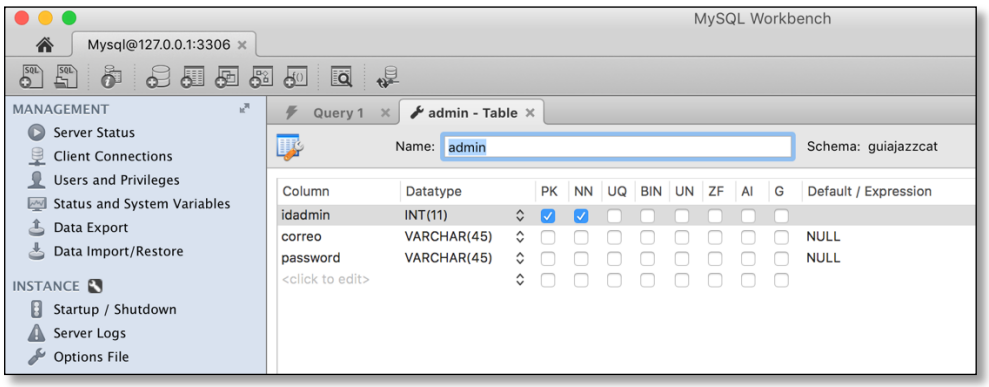


Figura 91 – Exemple MySQL Workbench

4.6.5 PHPMyAdmin

PHPMyAdmin és una eina de creació i gestió de bases de dades que ve incorporada en molts FrameWorks com el descrit anteriorment Codeigniter, es pot usar tant en local usant MAMP com a la majoria de servidors. Es tracta d'una eina gratuïta que permet importar arxius MySQL o crear directament la base de dades de forma senzilla.

Sense haver de codificar cap línia permet la creació de taules, inserció de dades, modificació, esborrat i moltes altres funcions així com a testejar *querys* (Veure Figura 93 i Figura 94). Requereix d'una autenticació per a mantenir un nivell de seguretat tal i com es veu a la Figura 92.

Figura 92 – Autenticació per l'accés a PHPMyAdmin

Figura 93 – Exemple estructura de Taules a PHPmyAdmin

Figura 94 – Estructura d'una taula concreta a PHPMyAdmin

5 Implementació

En aquest apartat s'expliquen els passos seguits per a la elecció de tecnologies, processos de disseny e implementació de la aplicació web, Web Service i aplicació mòbil.

5.1 Web

Amb els requisits descrits al document, s'ha descartat usar un CMS ja que l'aplicació a desenvolupar és molt específica i no s'ha trobat cap plantilla adhoc ni gratuïta ni de pagament que satisfaci tots els requeriments. És pot usar un CMS i desenvolupar el codi però la programació és més complexa al adaptar-se a les llibreries i funcions del CMS escollit.

Dins de les pàgines codificades s'han vist una gran varietat d'opcions. Per aquest projecte específic qualsevol d'elles hagués sigut una opció vàlida, llevat potser, de Javascript que està més orientat a les aplicacions d'una sola pàgina.

Al ser totes opcions vàlides, s'ha optat per triar una que tingui més presència aquests últims anys i una visió de futur atractiva en comptes d'altres més tradicionals com PHP i Java. Això deixa Perl, Ruby, Elixir, Go i Python com a possibles tecnologies a escollir.

Dins aquestes opcions s'ha optat per aquelles que tinguin més suport online, compatibilitat de llibreries i facilitin un desenvolupament més còmode. S'han analitzat les tendències de creixement i ús mostrades al tema 4 i s'han buscat els entorns de desenvolupament compatibles per a veure el seu funcionament i facilitat d'ús. Per tot això s'ha acabat triant Python amb el Framework Django i entorn de desenvolupament Pycharm.

Com ja s'ha explicat anteriorment ambdues tecnologies doten d'eines que faciliten molt la programació d'aplicacions web. És una tecnologia molt usada avui en dia per grans companyies com Instagram, Pinterest i New York Times. Les especificacions del projecte encaixen perfectament amb les característiques de Django ja que és ideal tant per l'aplicació web com per al Web Service (Django Rest) i al no tenir molta experiència personal ni amb el llenguatge ni amb el Framework, és un repte i una motivació. A més, Python és un llenguatge molt demanat al món laboral.

Escollida la tecnologia, els passos a seguir es divideixen en la etapa de disseny, instal·lació i configuració de l'entorn i desenvolupament.

5.1.1 Etapa de disseny

En capítols anteriors ja s'han descrit els objectius, estudiat la competència (estat de l'art) i especificat el projecte (ERS). Aquestes tres etapes son fonamentals en la etapa de disseny i marquen el camí a seguir en fases posteriors, de fet són necessàries per a conèixer l'abast, la viabilitat del projecte i permetre escollir la tecnologia i orientar el disseny.

La base de dades és un altre aspecte indispensable abans de començar el desenvolupament per a saber amb quines dades es tractarà. El disseny i la implementació de la base de dades s'expliquen més endavant en un apartat diferent.

Definits tots aquests aspectes cal entrar en etapa de disseny visual. Django segueix el patró de disseny MVT descrit anteriorment com a variant del MVC. La principal diferència de la variant amb el MVC és que Django, en aquest cas, s'encarrega de fer de controlador i gestionar el model i la vista i ens deixa el template, un fitxer HTML barrejat amb DTL, el llenguatge de template de Django.

Cal seguir un *responsive-design* així que com a punt de partida s'optarà per una plantilla de Bootsrap que defineix el menú de navegació i la estructura de columnes com a punts de partida. La pàgina tindrà diversos apartats en pàgines separades per tant s'ha optat per un menú de navegació vertical adaptable a la mida de pantalla on cada element del menú serà una secció de la pàgina.

Tenint clar els elements i contingut bàsic queda definir la disposició de la resta d'elements amb els que podrà interactuar l'usuari final. Per no programar i anar desfent canvis sinó acaba de quedar bé s'han fet servir *Mockups* i testos d'usuari per afinar la usabilitat i enteniment a la hora de disposar els elements i pàgines.

Els *Mockups* són una representació gràfica i molt bàsica del que acabarà sent la vista final amb la que interactuarà l'usuari. En aquest cas s'ha usat l'eina online mocflow.com, una eina que permet crear *mockups* a partir d'elements base com, per exemple, la plantilla de Bootsrap. (Figura 95)

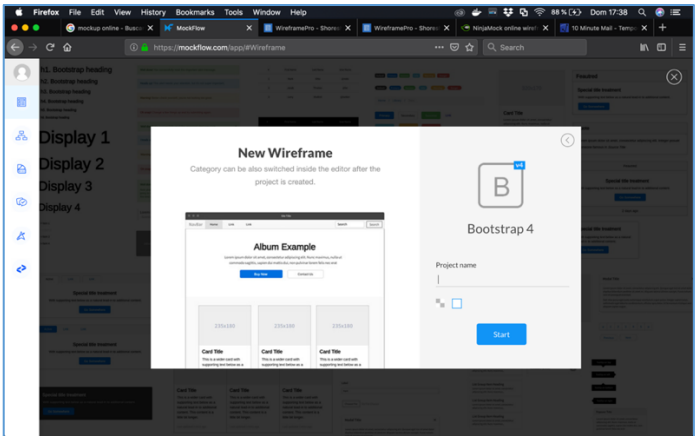


Figura 95 - Exemple eina MockFlow

Mitjançant els elements laterals podem afegir i modificar elements amb la funcionalitat de *Drag&Drop*.

Pel que fa a la tria de colors, al està relacionat amb el Surf i per tant amb el mar, s'ha optat per diverses tonalitats de blau combinat amb blanc per al text.

5.1.2 Instal·lació

Un cop es te una idea bastant clara de com serà la pàgina, es hora de començar a desenvolupar-la, però abans, cal instal·lar i configurar correctament tot l'entorn de treball. En aquest apartat s'explicaran els passos realitzats per dur a terme aquesta tasca en un Mac Os X. Prèviament s'ha explicat que es podia crear un servidor virtual amb la *localhost* i usant MAMP. En aquest cas, al usar Django, tot aquest procediment no és necessari, sinó que es crearà un entorn virtual on s'encapsularà el projecte.

Primer de tot, cal tenir instal·lar Python i pip. Pip un sistema de gestió de paquets de Python, és a dir, permet la seva instal·lació, actualització i eliminació. Amb Mac, es pot gestionar la instal·lació des del terminal i executant una sèrie de comandes com es veu a la Figura 96

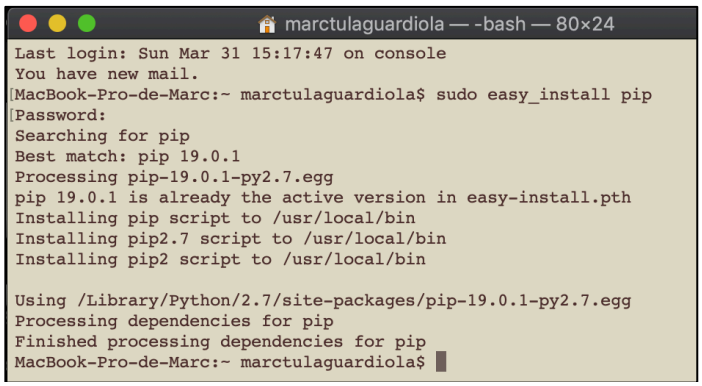


Figura 96 – Instal·lació PIP

Mac OS X per defecte porta instal·lat Python 2 però es pot instal·lar Python 3 des de la pàgina de descarregues de Python on s'haurà de instal·lar el *.pkg* seguint el típic assistent d'instal·lació. A les Figura 97 s'observa com es pot comprovar la versió de Python amb la comanda Python --version i python3 –version. Com a apunt, recalcar que no s'ha de desinstal·lar sota cap circumstància la versió de Python 2 en cas d'instal·lar la 3 ja que el sistema la usa.

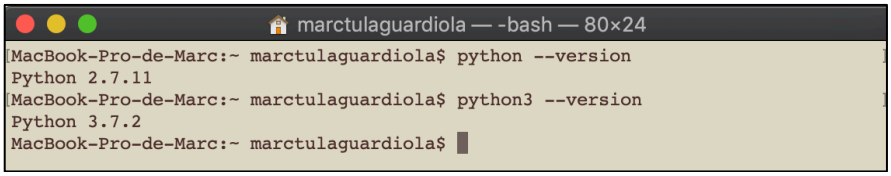


Figura 97 – Comprovar versió de Python

Un altre paquet que cal instal·lar a través del gestor PIP és l'entorn virtual, un ambient creat amb el propòsit de aïllar recursos com llibreries i entorn d'execució del sistema principal o altres entorns virtuals. A Python és comú usar diverses versions d'un mateix paquet que pot estar usant el Sistema Operatiu o altres projectes i per això s'aïllen els projectes en entorns virtuals. Per a instal·lar el paquet usarem la comanda: *sudo pip install virtualenv*

Un cop instal·lat el paquet, l'usem per a crear un entorn virtual. Aquest procediment es pot dur a terme des del terminal del sistema però en aquest cas, ho farem des del propi IDE per a felicitar la incorporació del entorn virtual al workspace on es troba el projecte.

L'entorn de desenvolupament escollit es PyCharms Comunity Edition (La versió gratuïta) de la companyia JetBrains. Al obrir-lo, cal crear un projecte seleccionant la versió de Python desitjada, es recomana la més actual i a continuació s'obra l'explorador d'arxius. A la part inferior es pot observar una consola amb el símbol de terminal des d'on es faran les operacions restants. A la Figura 98 s'observa l'escenari descrit.

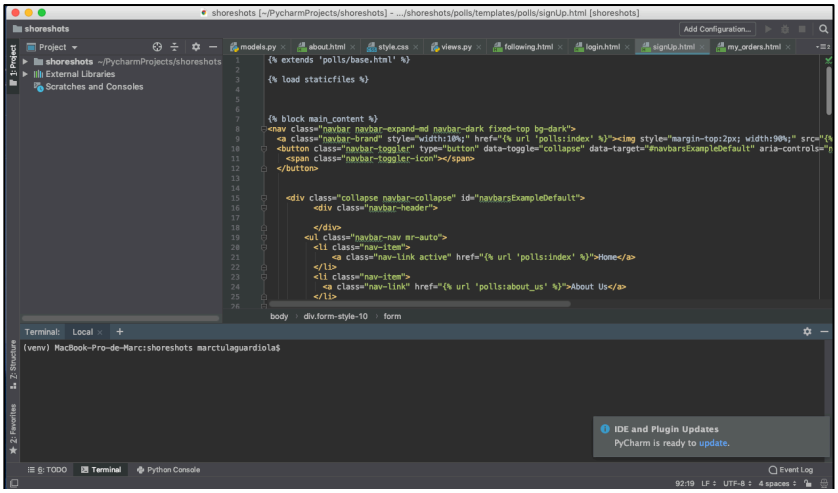


Figura 98 – PyCharm amb la consola a la part inferior

Des de la consola descrita crearem un entorn virtual amb la comanda: *virualenv -p python3 venv*, on venv fa referència al nom de l'entorn virtual creat. Al executar la comanda es pot observar al explorador de fitxers com apareix un nou directori anomenat venv tal i com s'observa a la Figura 99.

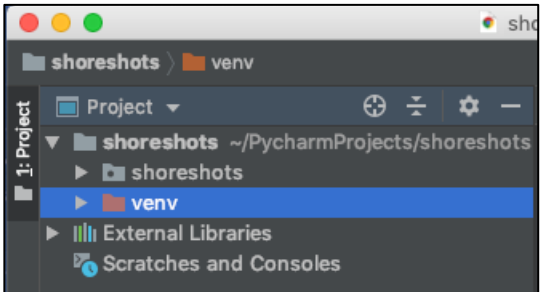


Figura 99 – Entorn virtual creat

Un cop creat, cal activar l'entorn amb la comanda: *source venv/bin/activate* i al executar s'observa com al terminal apareix (venv) davant del prompt de la consola per a indicar que el que s'instal·li, s'instal·larà a l'entorn virtual i no al sistema principal.

Tot i això, cal assegurar-se que realment estigui aquesta opció configurada a preferències->Project Interpreter verificar que està seleccionat l'entorn virtual creat tal i com s'observa a la Figura 100. A les captures anteriors ja es mostra aquest (venv) degut a que l'entorn ja estava activat però per defecte no apareix.

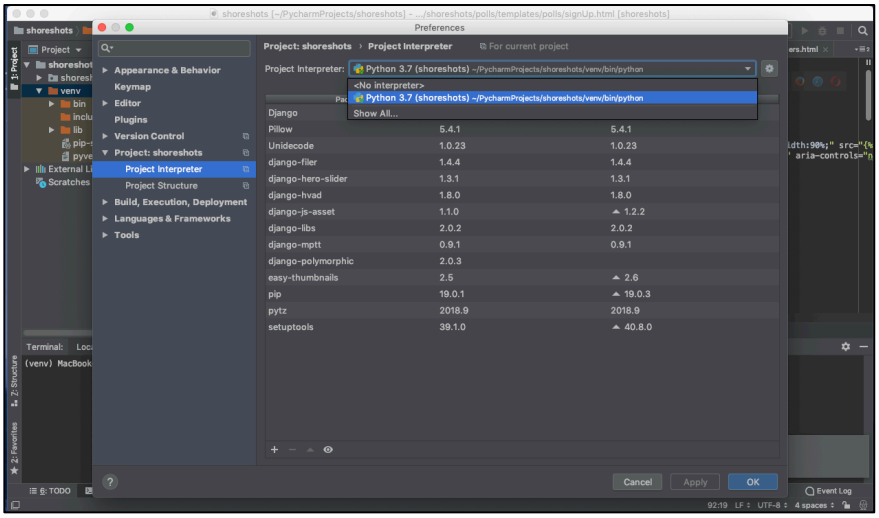


Figura 100 – Configuració Project Interpreter

Un cop creat, activat i configurat l'entorn de treball ja es pot procedir a la instal·lació de Django amb la comanda: *pip install django.*

5.1.3 Desenvolupament

5.1.3.1 Configuració

Amb l'entorn creat i Django instal·lat ja es pot procedir a la implementació. En aquest apartat es mostraran les diverses pàgines creades i s'analitzaran els elements més rellevants de *Backend.*

Per a començar el projecte cal introduir al terminal inferior: *django-admin startproject shoreshots*. On shoreshots en aquest cas és el nom que tindrà la pàgina. Immediatament es crea un nou directori amb el nom escollit tal i com s'observava a la Figura anterior 99 Observant el contingut d'aquest nou directori s'observen els següents fitxers:

- **_init_.py**: És un fitxer de configuració que indica que el directori es un paquet Python.
- **Settings.py:** Fitxer de configuració amb els paràmetres del projecte: Bases de dades instal·lades, aplicacions instal·lades, llenguatge, zona horària, etc.
- **Urls.py:** Fitxer que descriu les diferents urls que formen la pàgina web.
- **Wsgl.py:** És un fitxer per a la pujada del contingut creat a un servidor.
- **Manage.py:** Ens permet comunicar amb Django via terminal.

Arribats aquí, s'ha creat el projecte que engolava tota a pàgina web. Dins d'aquest projecte, es crearà ara, una aplicació que s'encarrega de gestionar un apartat de la web

amb certes funcionalitats. És a dir, es pot crear una aplicació per a gestionar el login i registre, una per al blog, etc.

Per a crear l'aplicació cal entrar al directori del projecte: *cd shoreshots* i executar la comanda Python *manage.py startapp polls*. On polls fa referència al nom de l'aplicació, en aquest cas s'ha posat polls perquè és el nom per defecte de la documentació de Django per a la primera aplicació.

Immediatament després d'executar la comanda es crea un nou directori amb el nom de l'aplicació al interior del projecte amb fitxers:

- **_init_.py:** Igual que abans, indica que el directori és un paquet de Python
- **admin.py:** Fitxer que ajuda a crear nous models a la pàgina d'administrador que es veurà més endavant.
- **apps.py:** Fitxer de configuració de l'aplicació creada.
- **models.py:** Conté la base de dades amb les taules i atributs.
- **tests.py:** Usat per a realitzar proves de funcionament.
- **views.py:** Són les vistes finals en HTML.

El primer pas un cop creat tot l'entorn seria crear la base de dades. Cal recordar, però que l'objectiu és que l'aplicació es connecti a un Web Service per a consultar la informació. Per tant, serà el Web Service qui contingui i interactuï amb la base de dades. Tot i així, a continuació es mostra un petit exemple de com es podria implementar amb Django.

Al fitxer models.py es creen les taules necessàries, com per exemple usuari i imatge amb els atributs que volem emmagatzemar. A la Figura 101 es pot veure un exemple amb les taules creades com a classes.

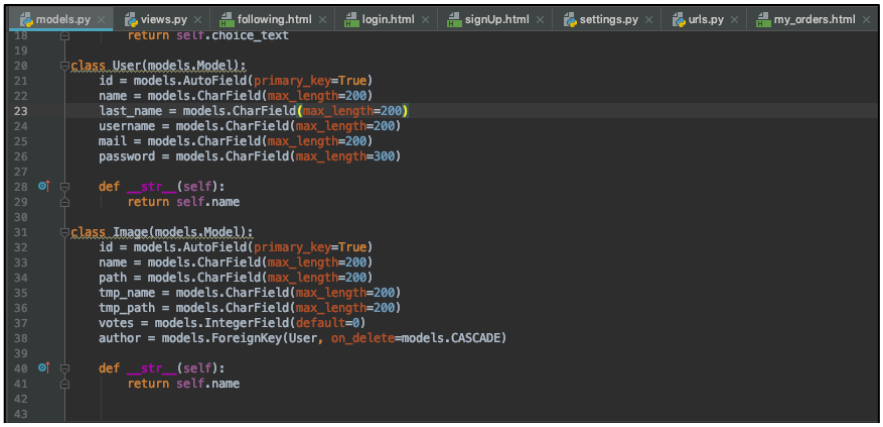


Figura 101 - Classes Image i User com a taules del model

Els canvis aplicats al fitxer models.py requereixen d'una migració amb una comanda de la consola per a poder ser finalment aplicats. La comanda per a realitzar aquesta operació és: *python manage.py makemigrations,* que vindria a ser com un commit on es preparen el canvis. Per a aplicar-los cal introduir una última comanda: *python manage.py migrate,* que vindria a ser com un push.

Quan es vulguin observar els canvis realitzats caldrà fer un *deploy* del servidor, a la consola inferior s'executa la comanda: *python manage.py runserver,* immediatament es podrà comprovar si existeixen errors ja que ens els marcarà la consola i en cas de no haver-hi s'observarà com comença la execució tal i com es mostra a la Figura 102.

```
(venv) MacBook-Pro-de-Marc:shoreshots marctulaguardiola$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
April 01, 2019 - 10:49:26
Django version 2.1.5, using settings 'shoreshots.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[01/Apr/2019 10:49:31] "GET /admin HTTP/1.1" 301 0
```

Figura 102 – Server Deploy

Mitjançant un navegador, es pot accedir a les urls descrites al fitxer urls.py del projecte. Per a gestionar la base de dades es pot accedir a la 127.0.0.1:8000/admin on el primer que ens demana es autenticar-nos. Cal crear un usuari amb permisos d'administrador, al terminal inferior de PyCharms s'introduiran les següents comandes.

Pyton manage.py createsuperuser. Aquesta comanda demana per pantalla la informació del usuari a crear com el nom, password, email, etc. Un cop creat l'usuari ja es pot accedir al panell d'administrador com es veu a les Figures 103 i 104.

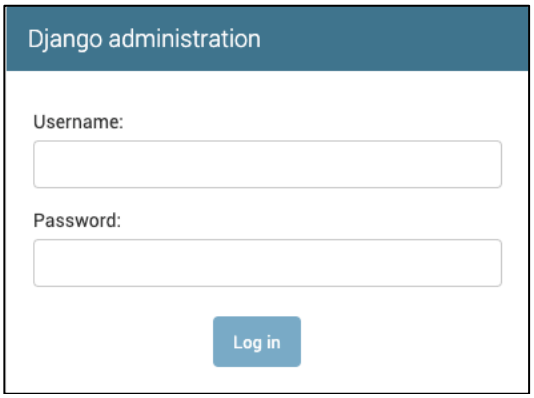


Figura 103 – Panell d'accés a l'administració de Django

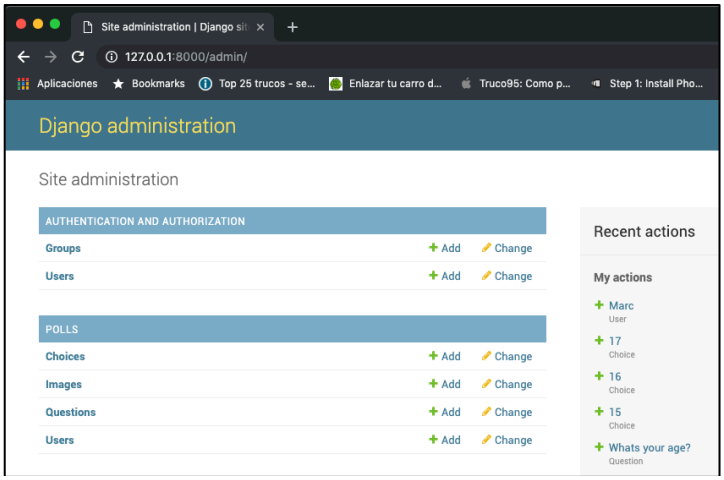


Figura 104 - Panell d'administració Django

Des de la pàgina d'administrador es poden gestionar els usuaris i grups que poden gestionar el projecte així com la base de dades. Es poden modificar les dades de les taules així com afegir-ne de noves, però per crear noves taules cal usar les comandes descrites anteriorment.

Per accedir a la pàgina s'ha de crear una nova url per a l'aplicació i indicar-la al fitxer de urls del projecte així com registrar l'aplicació, en aquest cas *polls,* al fitxer Settings.py del projecte tal i com es mostra a la Figura 105 i 106

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url, include

urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'polls/', include(('polls.urls', 'polls'), namespace='polls')),

]
```

Figura 105 – fitxer urls amb la url a l'aplicació

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'polls',
]
```

Figura 106 – Aplicacions registrades a Settings.py

Per a començar a crear contingut cal crear un nou fitxer python a l'aplicació que s'anomeni també urls.py. En aquest fitxer s'afegeixen les urls de les noves pàgines que es van creant. A la Figura 107 es mostren algunes de les urls creades.

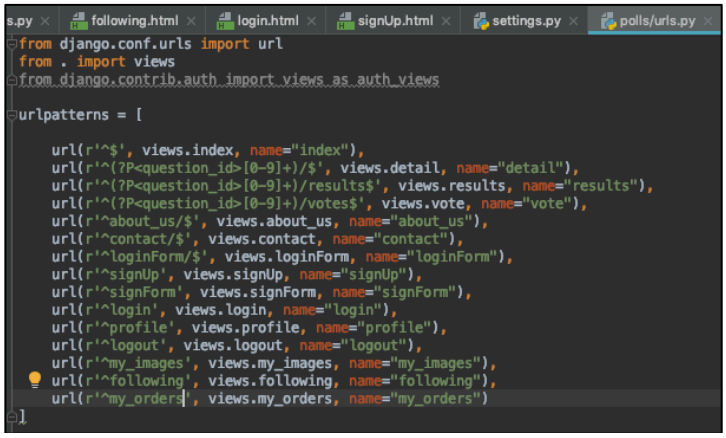


Figura 107 – Fitxer urls.py de l'aplicació creada

Cada url registrada defineix també, el nom d'una funció a executar al fitxer views.py. En aquest últim fitxer cal definir les funcions necessàries per cada url. Si cal tractar dades, o realitzar operacions abans de mostrar la vista és dins la funció on es realitzaran, per tant, actua com una mena de controlador. Seguidament es carrega la vista necessària. A la Figura 108 es mostra un exemple de fitxer views.py.

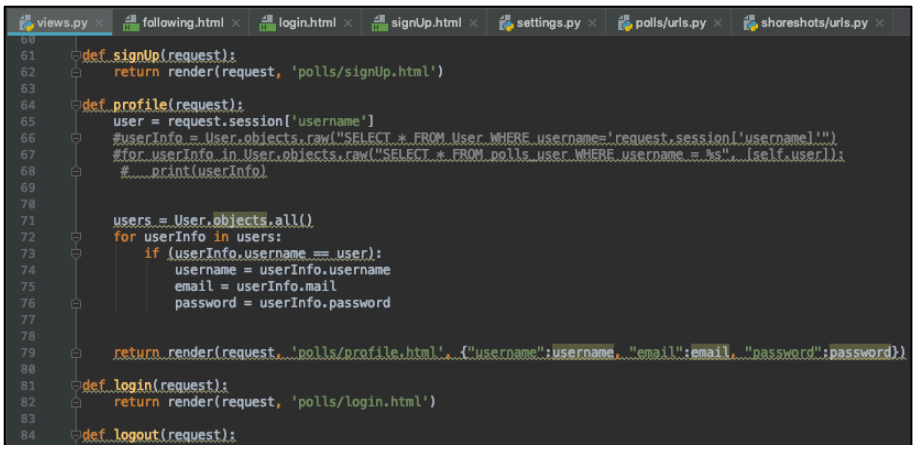


Figura 108 – Fitxer views.py

Per a crear les vistes finals, s'ha creat un directori *templates* i en el seu interior un altre anomenat *polls*. Tal i com s'ha explicat abans, es poden tenir diverses aplicacions registrades que s'encarreguen d'una secció de la web, amb aquesta estructura es separen les vistes de cada aplicació.

Dins d'aquest nou directori ja es pot crear un nou fitxer amb extensió html. Un dels aspectes rellevants de Django es que permet crear mòduls per a reaprofitar codi. En el cas de les vistes, es voldrà que totes mantinguin la mateixa aparença pel que fa al menú de navegació superior. Per a dur-ho terme es crea un fitxer *base.html* amb el codi del menú de navegació i s'inclou el fitxer a la resta de vistes on es vulgui mostrar.

A la Figura 109 es mostra un exemple de la estructuració de fitxers.

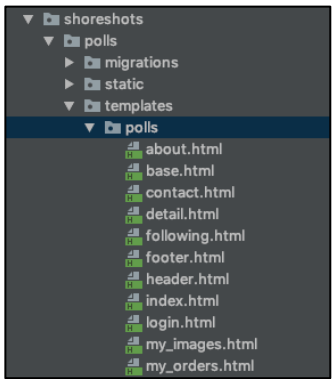


Figura 109 – Estructura de fitxers

A la Figura 110 es mostra un exemple del fitxer base.html i com incloure'l a altres vistes.

```
<!DOCTYPE html>
{% load staticfiles %}


<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="Marc Tula, Surf And Share">
    <meta name="generator" content="Jekyll v3.8.5">
    <meta charset="UTF-8">
    <title>Surf And Share</title>
    <link rel="stylesheet" type="text/css" href="{% static 'polls/bootstrap-4.2.1-dist/css/bootstrap.min.css' %}"/>
    <link rel="stylesheet" type="text/css" href="{% static 'polls/style.css' %}"/>
    <link rel="stylesheet" href="{% static 'polls/font-awesome-4.7.0/css/font-awesome.min.css' %}">
    <script src="{% static 'polls/jquery-3.3.1.min.js' %}"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
    <script src="{% static 'polls/main.js' %}"></script>
    <link rel="shortcut icon" href="{% static 'polls/images/favicon.ico' %}" type="image/x-icon">
    <link href='http://fonts.googleapis.com/css?family=Bitter' rel='stylesheet' type='text/css'>
    <!--<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">-->

</head>
```

Figura 110 – Fitxer base.url

```
{% extends 'polls/base.html' %}

{% load staticfiles %}



{% block main_content %}
<nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <a class="navbar-brand" style="width:10%;" href="{% url 'polls:index' %}"><img style="margin-top:2px; width:90%;"
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-cont
    <span class="navbar-toggler-icon"></span>
  </button>


    <div class="collapse navbar-collapse" id="navbarsExampleDefault">
          <div class="navbar-header">

          </div>
        <ul class="navbar-nav mr-auto">
          <li class="nav-item">
              <a class="nav-link active" href="{% url 'polls:index' %}">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'polls:about_us' %}">About Us</a>
          </li>
```

Figura 111 – Inclusió de base.url a altres vistes.

A la Figura 111, per a incloure el fitxer s'afegeix la línia *'{extends 'polls/base.html' %}',* usant el llenguatge DTL (Django Template Language).

A continuació es descriuen els elements més rellevants de les pàgines principals de la pàgina web.

5.1.3.2 Home

El Home serà la pagina inicial que es mostrarà als usuaris. Per a facilitar la cerca d'imatges als usuaris que ja coneguin la pàgina, el primer que es mostra és una barra de cerca on es poden introduir paraules clau com el fotògraf, Localització o data. Al mateix *banner* es destaquen les seccions de nou fotògraf per a permetre un registre ràpid, imatges de l'última temporada, i les imatges millor valorades.

Seguidament es troba una descripció sobre l'activitat de la pàgina i de Shoreshots. Com que el text encara està per a definir s'ha usar un generador *Lorem Ipsum.* D'aquesta manera i sense necessitar cap interacció via clic per part de l'usuari, amb un petit *scroll* un usuari ja troba la informació necessària per a entendre de que tracta la pàgina

El següent apartat mostra tres seccions per a la cerca d'escoles, localitzacions o lloguer de material. L'objectiu és el de poder oferir més serveis que els competidors no només amb la venta d'imatges sinó buscant convertit Shoreshots en un portal de referència del surf per a que la gent pugui trobar informació sobre on anar a practicar el esport, quines són les millors escoles o campus per aprendre o llocs on poder llogar material.

Per últim, abans d'arribar al *footer* i al final de pàgina, apareixen les ultimes imatges de fotògrafs. La imatge haurà de dur una marca d'aigua, però per aquesta primera versió de proba de moment s'incorpora la imatge normal. Es crida a una funció del Web Service que retorna la informació de les 6 últimes imatges (nom, fotògraf, localització i data).

Per al tractament de les imatges existeixen diverses opcions, la primera es deixar llibertat al fotògraf pel que fa a les mides de la imatge, i per tant mostrar-les totes en diferents mides.

S'ha optat per a posar una restricció de mida i així mostrar totes les imatges igual. Les dades introduïdes no son reals sinó d'entorn de proves amb dades fictícies. Les imatges s'han tret de la pàgina web pexels.com les quals es poden compartir i descarregar.

A la Figura 112 es mostra el resultat d'una cerca i a la Figura 104 l'aspecte general de la pàgina Home

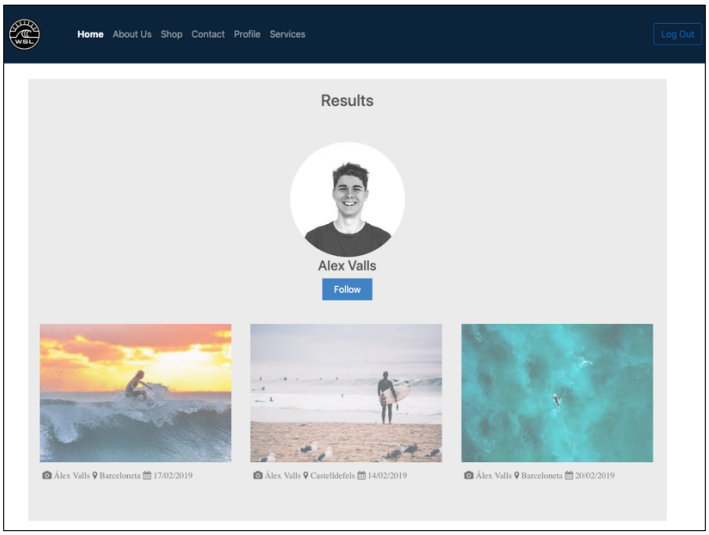


Figura 112 – Resultats d'una cerca des de la barra del home

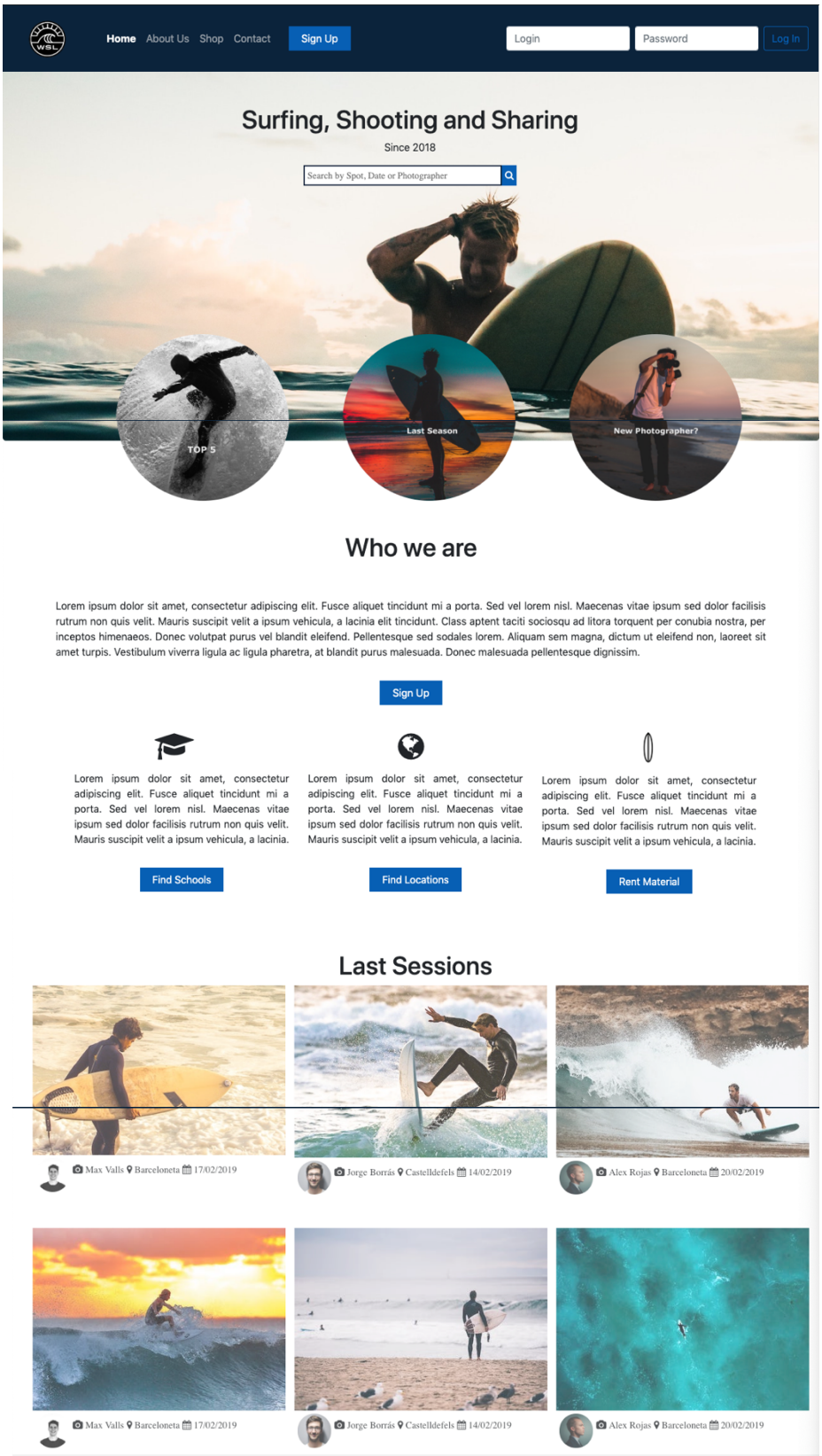


Figura 113 - Home

Una altre de les característiques del home i de la resta de pàgines és que comprova si hi ha algun usuari en sessió. En cas afirmatiu, el menú de navegació a carregar varia, mostrant una pàgina de perfil en comptes del formulari de login. A la figura 114 es mostra el menú amb un usuari en sessió.

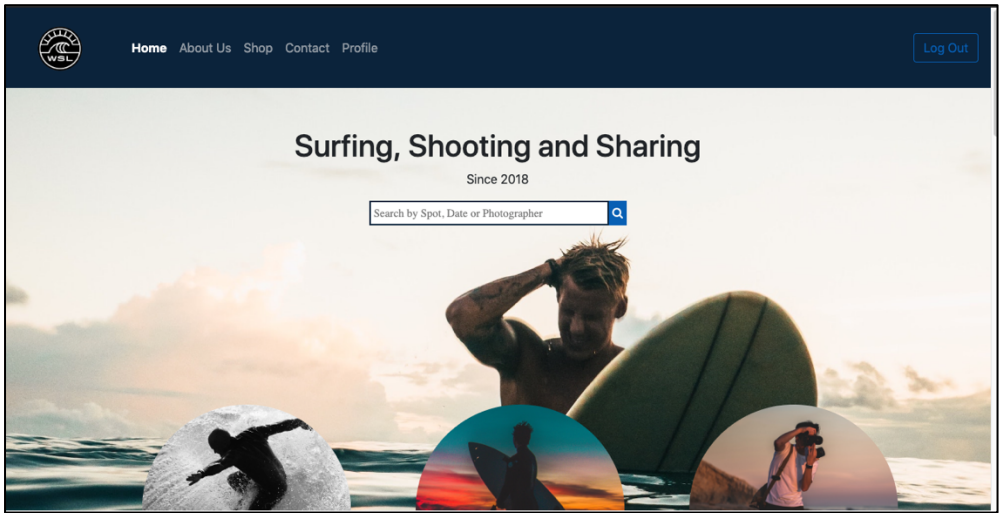


Figura 114 - Home amb usuari en sessió

Per a implementar les sessions s'usen les pròpies eines de Django. Al validar el formulari de login d'un usuari, s'inserta el nom d'usuari en sessió tal i com es mostra a la Figura 115. Al inserir el nom d'usuari, cal estar segurs de que aquest nom és únic, per això al formulari de registre es comprova que el nom no existeixi abans de procedir amb el registre.

```
for u in users:
    if u.username == username:
        usernameok = 1

    if u.password == password:
        userpassok = 1

if usernameok and userpassok:
    request.session['username'] = username

    user = request.session['username']
    users = User.objects.all()
    for userInfo in users:
        if (userInfo.username == user):
            username = userInfo.username
            email = userInfo.mail
            password = userInfo.password

    return render(request, 'polls/profile.html', {"username": username, "email": email, "password": password})
```

Figura 115 – validació i inserció usuari en sessió

Abans de carregar caldrà validar si la variable request.session[‘username’] està plena o buida per a saber quin menú de navegació cal carregar tal i com es mostra a la Figura X.X. Concretament s'usa el llenguatge DTL per a realitzar la comprovació: *{% if session == “notset” %}* (Figura 116)

```
{% block main_content %}

{% if session == "notset" %}
    <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
      <a class="navbar-brand" style="width:10%;" href="{% url 'polls:index' %}"><img style="margin-top:2px; width:90%;"
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-cont
        <span class="navbar-toggler-icon"></span>
      </button>
```

Figura 116 - Comprovació de si existeix sessió

Amb el botó que de *Log Out* que es mostra a la Figura 114 es tanca la sessió i es torna a carregar el Home. Per a eliminar la sessió s'escriu en Python des del fitxer views.py:

```
del request.session['username']
```

5.1.3.3 – Pàgina de Login

Per tal d'accedir al sistema, l'usuari pot introduir les credencials des del propi menú de navegació o bé a la pàgina de login que es mostra a la Figura 117.

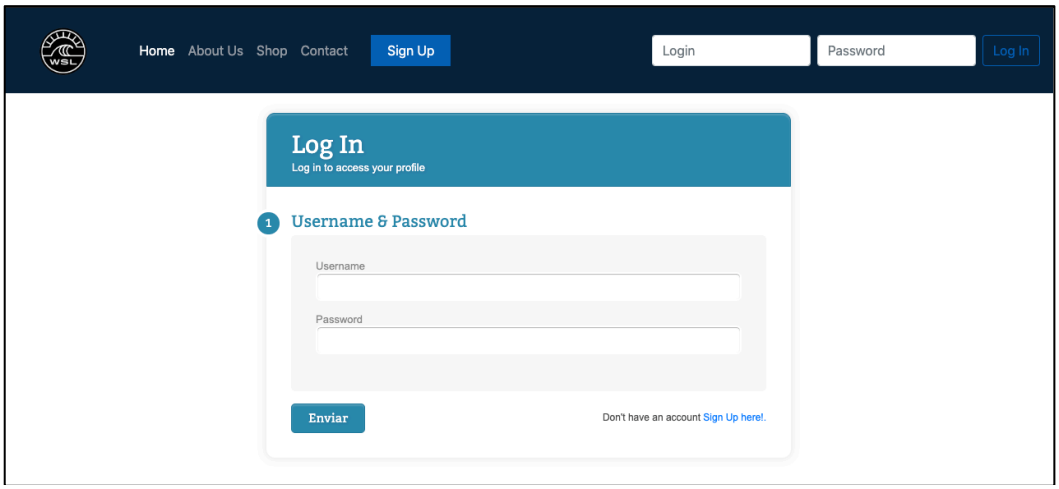


Figura 117 – Pàgina de Login

El formulari incorpora un enllaç per als usuaris que no tinguin un compte registrat. Per a la validació es recuperen tots els usuaris de la base de dades via Web Service i en bucle es comprova si el nom d'usuari i password coincideixen.

En cas afirmatiu es crea la sessió i es mostra la pàgina de perfil d'usuari amb la seva informació personal i el menú de navegació en sessió tal i com es mosta a la Figura 118.

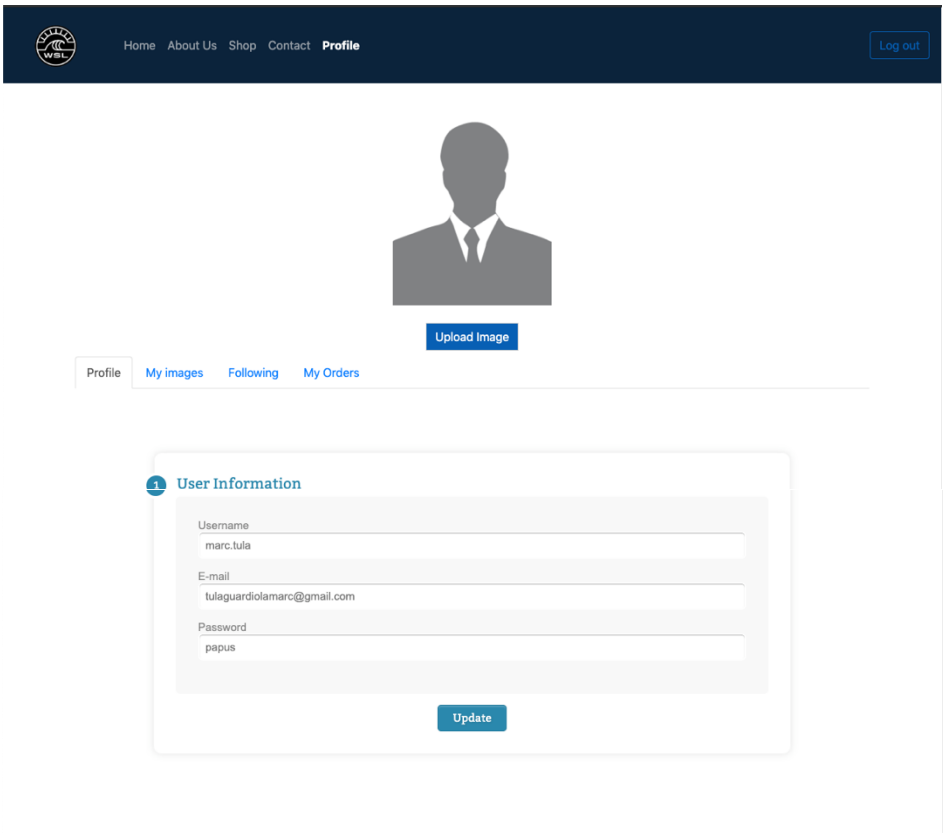


Figura 118 – Pàgina de perfil d'usuari autenticat

La informació es recupera de la base de dades via Web Service i s'envia a la vista a través del fitxer Views.py. Des de la vista s'accedeix a la informació amb llenguatge DTL com es veu a la Figura 119

```
<div class="form-style-update">
    <form action="" method="post">
        <div class="section"><span>1</span>User Information</div>
        <div class="inner-wrap">
            <label>Username <input type="text" name="username" placeholder="{{username}}"/></label>
            <label>E-mail <input type="email" name="mail" placeholder="{{email}}"/></label>
            <label>Password <input type="password" name="password"  placeholder="{{password}}" /></label>
        </div>
        {% csrf_token %}
        <div class="button-section">
         <center><input type="submit" name="Update" value="Update" /></center>
        </div>
    </form>
</div>
```

Figura 119 – Llenguatge DTL per al accés a la informació

En cas de no validar el login es mostrarà un missatge per pantalla conforme el nom d'usuari o la contrasenya introduïts no són correctes i es torna a mostrar la pàgina tal i com es mostra a la Figura 120.

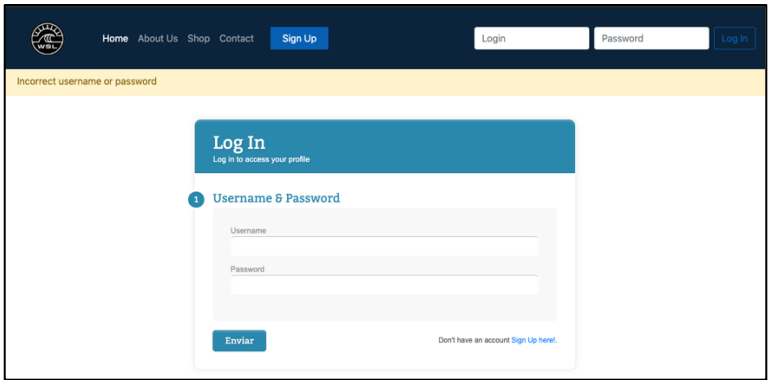


Figura 120 – Login incorrecte

5.1.3.3 Registre

El registre demana informació necessària com el nom, correu, contrasenya, tipus (Surfista o fotògraf) i els envia al Web Service per a la seva inserció. Prèviament es valida que el correu o el nom d'usuari no existeixin ja a la base de dades. A la Figura 121 es mostra el formulari de registre.

El formulari valida que al camp de correu s'introdueixi un correu vàlid, que les contrasenyes coincideixin i que s'acceptin els terminis de política de privacitat. A la part superior s'observen dues icones interactives per a que l'usuari esculli a quin tipus de perfil es vol registrar, fotògraf o surfista (Figura 122).

Figura 122 – Opcions interactives per a escollir el tipus de perfil a registrar

Per a donar aquest aspecte d'interacció s'ha usat Javascript com s'observa la Figura 123

```
<script>
        $(".checker").on("click", function(){
            $(".photoChecker").css("background-image", "url('/static/polls/images/photographer-icon.png')");
            $(".checker").css("background-image", "url('/static/polls/images/surfer_hover_icon.png')");

        });
        $(".photoChecker").on("click", function(){
            $(".checker").css("background-image", "url('/static/polls/images/surfer-icon.png')");
            $(".photoChecker").css("background-image", "url('/static/polls/images/photographer-hover-icon.png')");
        });
    </script>
```

Figura 123 – Javascript per a crear efectes d'interacció

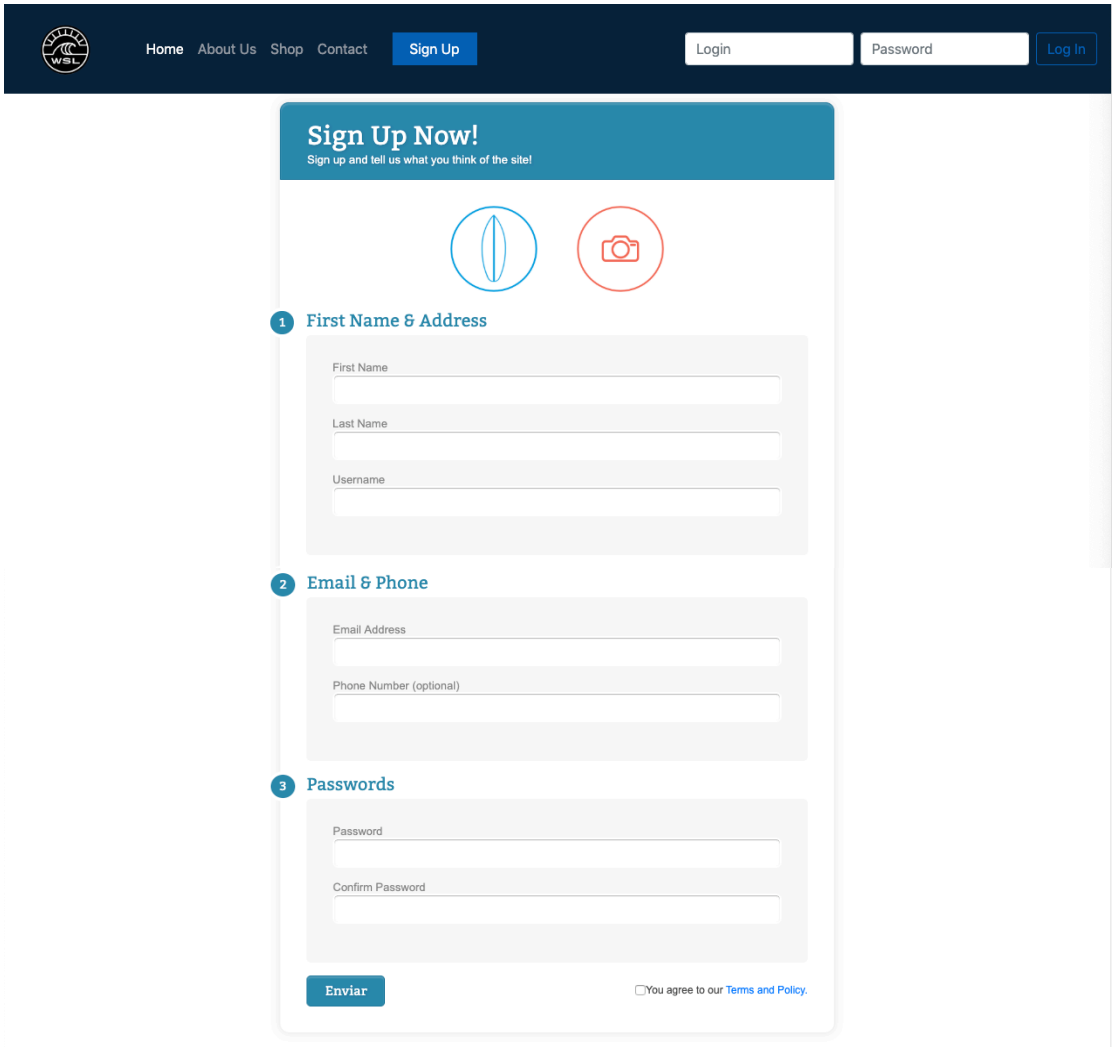


Figura 121 – Pàgina de registre

Per a realitzar la inserció es recuperen les dades al fitxer Views.py via *POST* i s'envien al Web Service com a paràmetres en format Json de la funció post. El Web Service introdueix les dades mitjançant un *serializer* com s'explica més endavant.

```
serialized_user ={
   "user": {
      "name":user.name,
      "lastName":user.last_name,
      "username":user.username,
      "email":user.email,
      "password":user.password,
      "type":user.type
   }
}
response = requests.post('http://127.0.0.1:8000/api/users/',json=serialized_user)
```

5.1.3.4 Pàgina de Perfil

Tornant a la pàgina de perfil, l'usuari podrà modificar la seva imatge de perfil que apareixerà a la part inferior de les imatges que pengi. A la part inferior de la pàgina, disposarà d'un menú de pestanyes on podrà veure i actualitzar la seva informació personal, veure les últimes imatges penjades (fins un màxim de 10) i penjar-ne de noves, consultar les compres realitzades i les persones a les que segueix (Surfistes o fotògrafs). (Figura 124)

La restricció de mostrar 10 imatges màxim és temporal degut a espai de servidor. A mesura que el projecte agafi a usuaris, a les línies del futur del projecte, es plantejarà l'ampliació de les imatges a mostrar i l'ampliació de l'espai amb nous servidors.

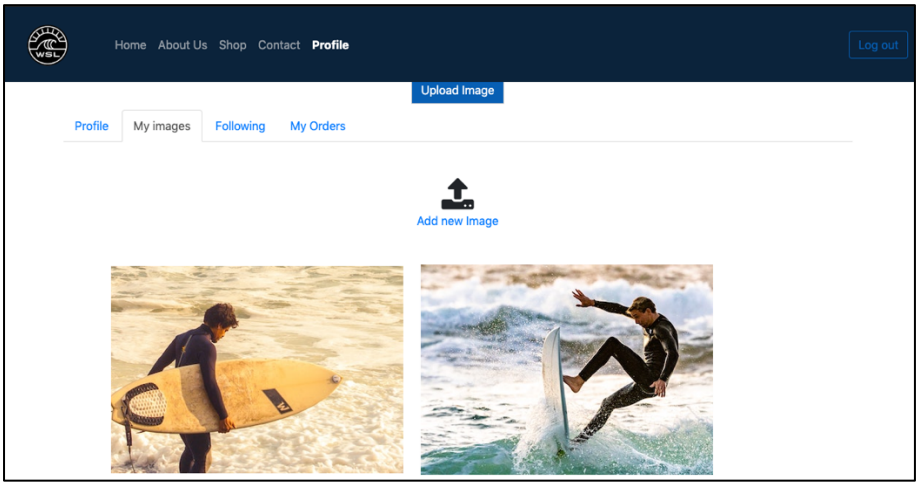


Figura 124 – Imatges penjades des d'un perfil

El menú de pestanyes s'ha realitzat usant classes predefinides de Bootstrap importat al projecte des de l'inici. Per a mostrar imatges es crida una funció del Web Service que rep el nom d'usuari com a paràmetre i recupera la informació necessària per trobar la imatge a la ruta que correspongui.

Pujar Imatges

A continuació es descriu el mecanisme usat per a la pujada d'imatges tant de surf com la de perfil d'usuari. La gestió de fitxers i directoris locals per part de Django no està configurada per defecte, el primer pas és modificar les fitxers de configuració per establir el directori on es pujaran les imatges.

- Es modifica el fitxer Settings.py afegint:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

- Es crea el formulari amb una nova url per a la seva gestió. La url s'afegeix al fitxer urls.py i una nova funció que recuperarà les dades al fitxer views.py. El formulari haurà de tenir un camp *input file*, el CSRF token i el camp enctype='multipart/form-data'.

- Es necessari importar les llibreries que permeten treballar amb fitxers i directoris:

```
from django.core.files.storage import FileSystemStorage
import os
```

- Es recupera la informació des de la funció del fitxer views.py i es guarda amb el nom desitjat, en aquest cas *profile.png*

```
if request.method == 'POST':
    upload_file = request.FILES['document']
    fs = FileSystemStorage()
    fs.save("profile.png", upload_file)
```

A la següent Figura 125 es mostra el canvi d'imatge de perfil un cop seguits els passos anteriors. La pàgina de perfil mira si existeix una imatge amb el nom profile.png al directori de l'usuari creat amb el registre. En cas satisfactori carrega aquesta imatge, sinó carrega la imatge per defecte mostrada a la Figura 118. Al pujar una nova imatge de perfil, primerament es cerca si existeix una antiga, la qual s'esborra per a que només existeixi una única imatge.

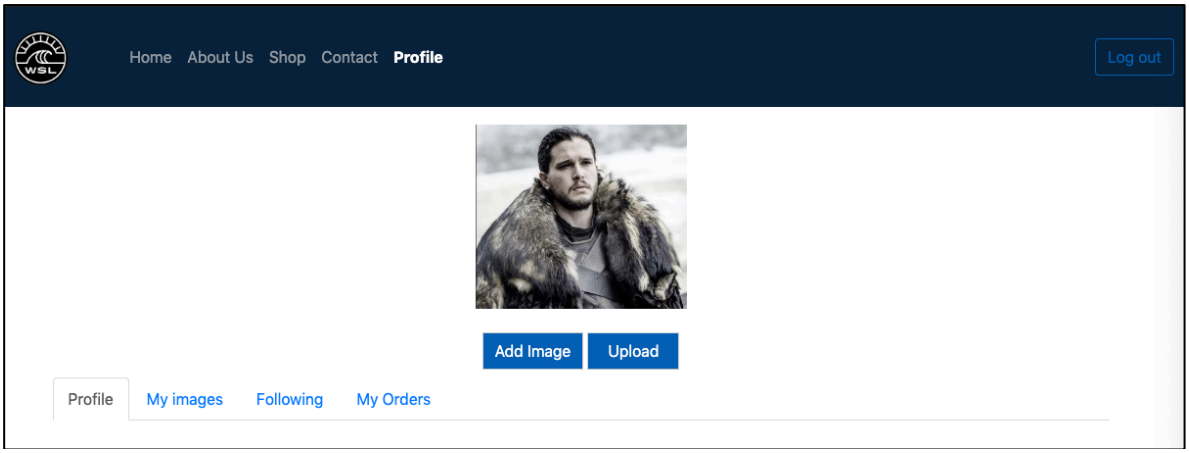


Figura 125 - Nova imatge de perfil

5.1.3.5 Pàgina de contacte

La pàgina de contacte (Figura 126) mostra localització, telèfon i correu per a contactar en cas d'algun problema, dubte o suggeriment per part dels usuaris. Seguidament mostra un formulari de contacte on es facilita el contacte. Per últim es mostra el mapa de google maps on trobar la direcció.

Per a realitzar les cartes d'informació s'han usat les classes de bootstrap per a dividir l'ample de pantalla en columnes. S'ha afegit un afecte css (Zoom in) al passar el ratolí (Hover) per a destacar l'element que s'està observant de la resta i donar aspecte de pàgina interactiva.

El mapa s'ha introduït després de generar-lo des de google maps i obtingut el *iframe* corresponent com es mostra a la Figura 127

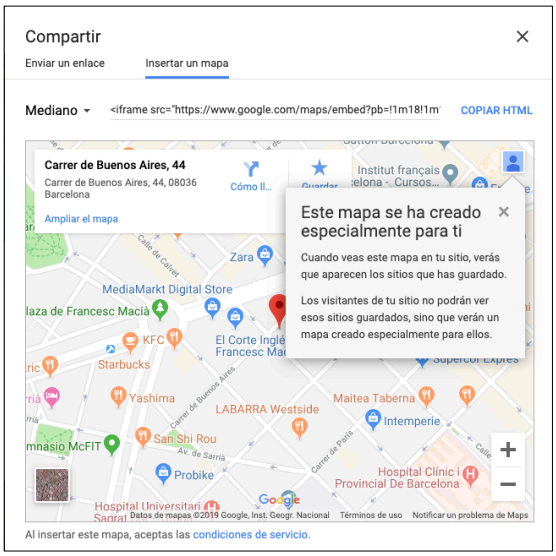


Figura 127 – Generar Google Maps

Per a poder enviar correus cal realitzar alguns canvis al fitxer Settings.py introduint les dades del servidor de correu on està allotjada la web. En aquest cas, el servidor és el *localhost*, per tant, es pot usar el servidor SMTP que Django incorpora. Al fitxer Settings.py cal afegir el codi que s'observa a la Figura 128.

```
if DEBUG:
    EMAIL_HOST = 'localhost'
    EMAIL_PORT = 1025
    EMAIL_HOST_USER = ''
    EMAIL_HOST_PASSWORD = ''
    EMAIL_USE_TLS = False
    DEFAULT_FROM_EMAIL = 'testing@example.com'
```

Figura 128 – Configuració servidor SMTP de Django

Tot i això, el servidor SMTP de Django és limitat, per això s'ha usat el servidor de Google. S'ha creat un compte per a gestionar la aplicació, canviant les dades mostrades a la Figura 128 per les corresponents del compte Google.

Al fitxer views.py s'han recuperat les dades del formulari via POST, importat la llibreria: *from django.core.mail import EmailMessage* i el codi que es mostra a la Figura 129.

```
def sendMail(request):

    name = request.POST.get('name')
    mail = request.POST.get('email')
    message = request.POST.get('message')

    email = EmailMessage("Contact From Web", name+message, to=['shoreshots.info@gmail.com'])
    email.send()
```

Figura 129 - Enviament mail de contacte.

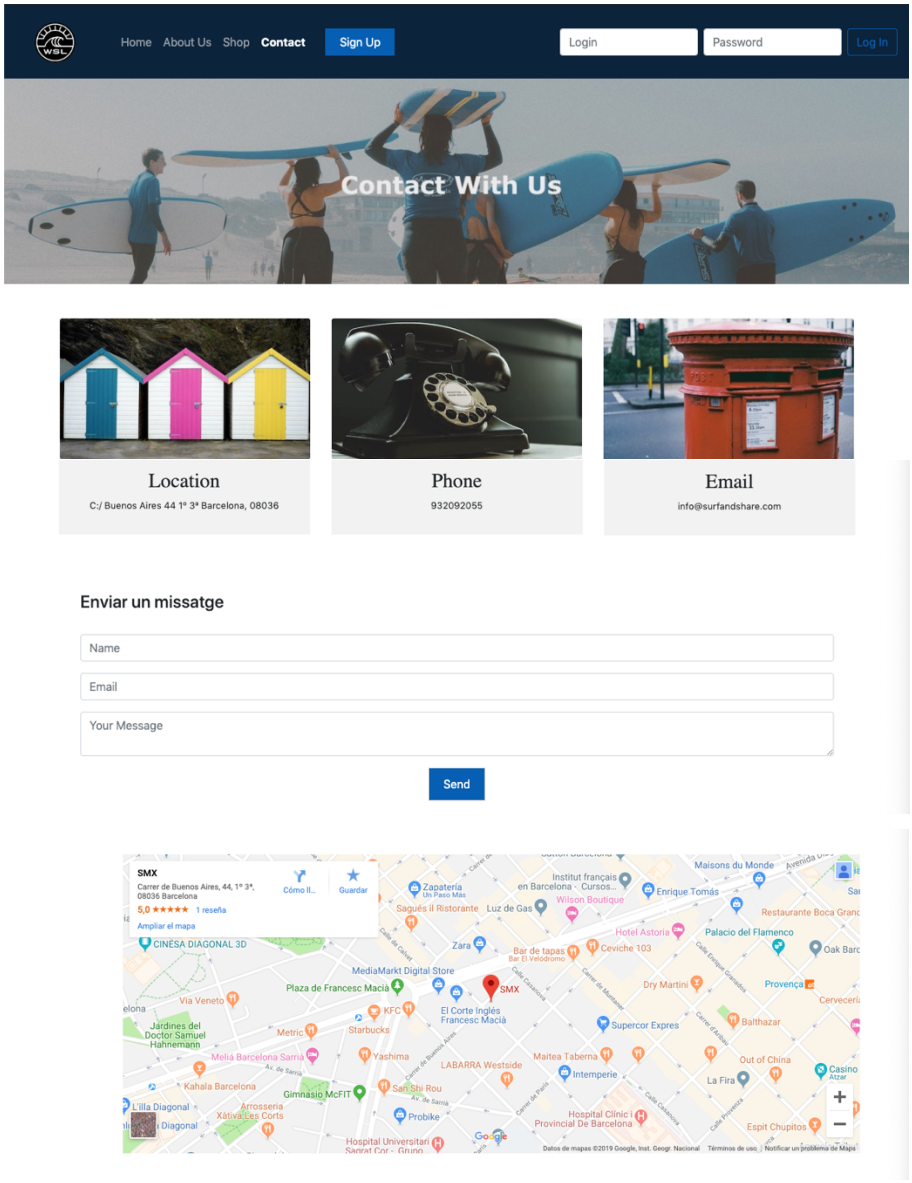


Figura 126 – Pàgina de contacte

5.1.3.6 About us

La pàgina d'about us que es mostra a la Figura 130 conté informació per a que l'usuari conegui de forma ràpida l'activitat i història de la empresa. Es concret, es mostra un text amb aquesta informació seguit de les descripcions dels serveis principals (compra i pujada d'imatges), les millors imatges, informació sobre el nombre d'usuaris registrats, nombre d'imatges pujades, etc. i col·laboradors.

S'ha usat bootstrap per a la divisió de l'ample de pantalla en columnes i JavaScript per a la creació dels sliders que van passant les imatges de forma automàtica. Les dades i textos introduïts son provisionals, així com les imatges dels col·laboradors i informació dels registres i imatges pujades.

Per a mostrar un nombre correcte d'usuaris registrats i imatges pujades/comprades, s'usarà una funció del Web Service, la qual retornarà el nombre de files de la taula d'usuaris i d'imatges. Per a retornar el nombre de files s'usarà la query: *SELECT COUNT(*) FROM users;*

A la Figura 131 es mostra el codi Javascript usat per a la creació dels visors d'imatges interactius que passen les imatges de forma automàtica.

```
var $slider = $('#slider');
var $slideContainer = $('.slides', $slider);
var $slides = $('.slide', $slider);

var interval;

function startSlider() {
    interval = setInterval(function() {
        $slideContainer.animate({'margin-left': '-='+width}, animationSpeed, function() {
            if (++currentSlide === $slides.length) {
                currentSlide = 1;
                $slideContainer.css('margin-left', 0);
            }
        });
    }, pause);
}
function pauseSlider() {
    clearInterval(interval);
}

$slideContainer
    .on('mouseenter', pauseSlider)
    .on('mouseleave', startSlider);

startSlider();
```

Figura 131 - Codi Javascript per als visors d'imatges.

Per als colors de fondo, s'ha usat la pàgina HTML Color codes per a escollir i provar els diferents colors disponibles. La pàgina genera un codi de color hexadecimal que s'introdueix a la propietat css: *background-color: #424242*

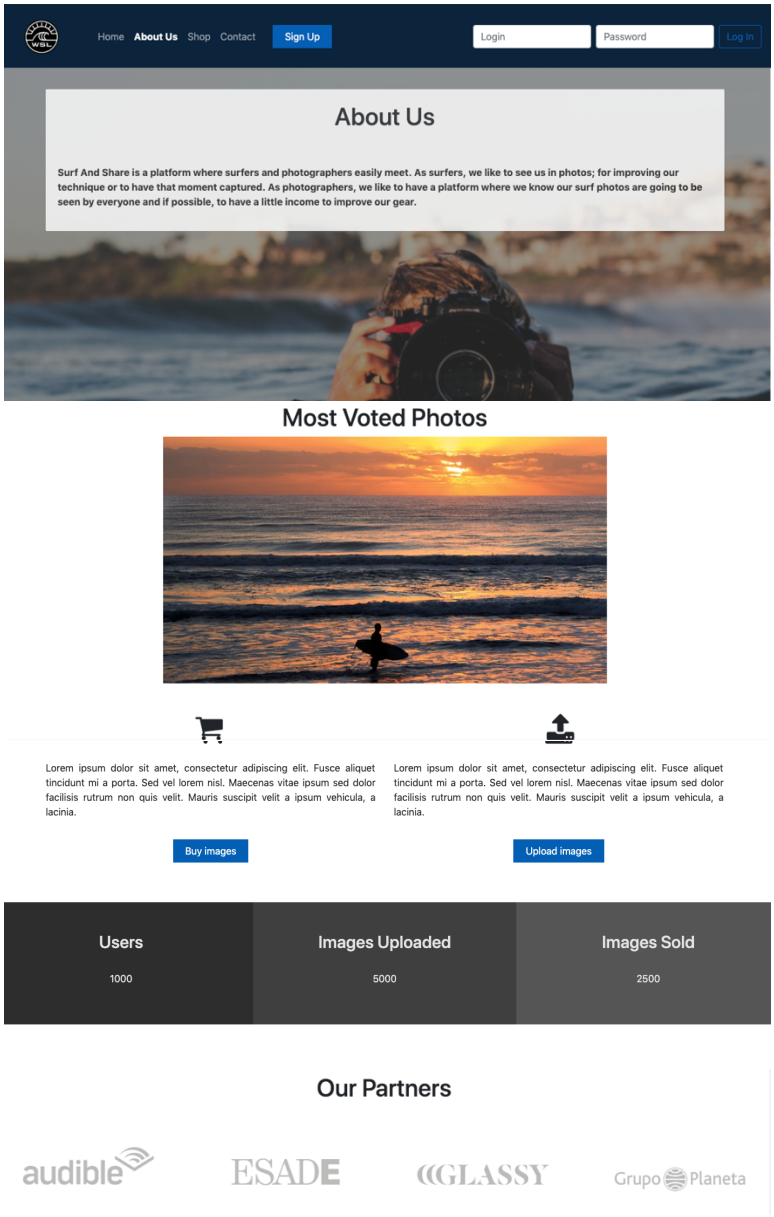


Figura 130 - Pàgina About Us

5.1.3.7 Serveis

Com ja s'ha explicat, Shoreshots vol convertir-se en una marca de referència en el món del surf i no només vendre imatges online. Per això, ofereix una pàgina de serveis

relacionats on l'usuari podrà buscar noticies recents, localitzacions on practicar l'esport, escoles i campaments per a aprendre i llocs on llogar material.

Per a mostrar aquests serveis d'una manera visual i atractiva s'ha usat una nova plantilla de Bootstrap. A la Figura 132 es mostra la pàgina principal de serveis.

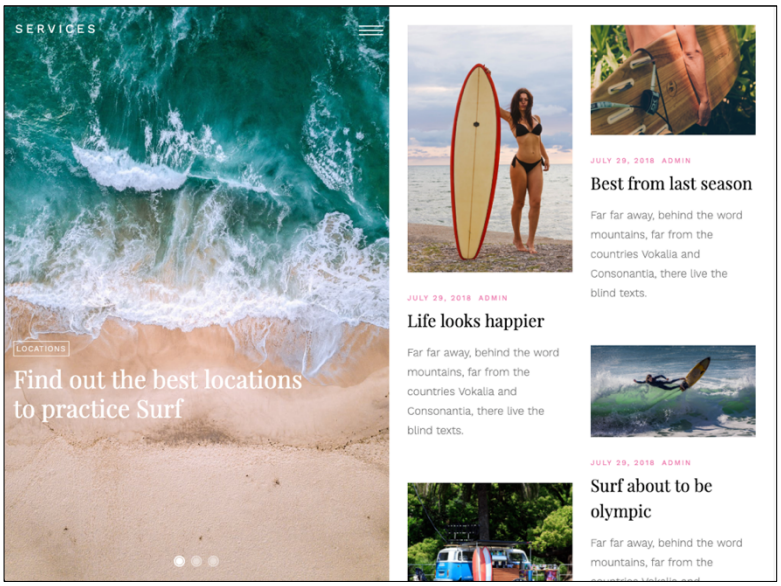


Figura 132 – Pàgina principal de Serveis

A la Figura superior s'observa un format molt més estil blog que ens permet cercar noticies de forma ràpida i visual. Mitjançant el botó superior es desplega un menú amb les diferents opcions, entre elles tornar a la pàgina principal.

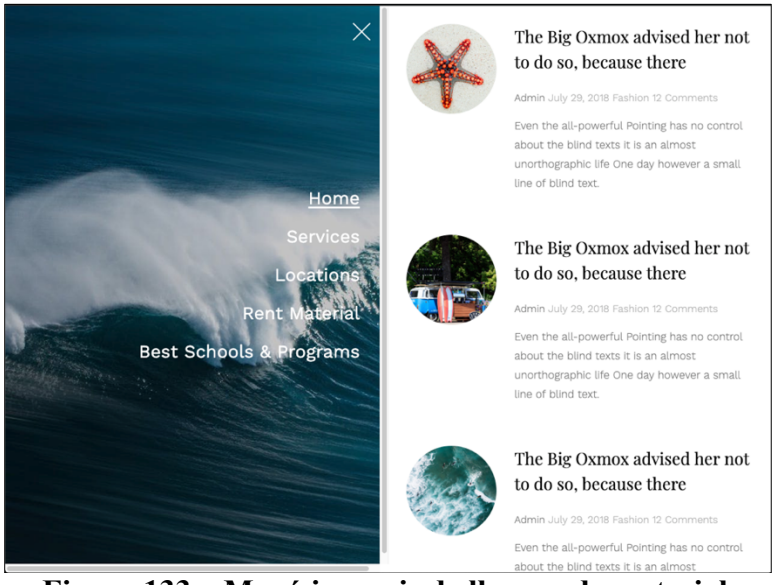


Figura 133 – Menú i serveis de lloguer de material

A mesura que seleccionem una categoria (Lloguer de material, escoles i càmpings, etc.) canvia l'estil en que es mostra la informació per a ajudar al usuari a discernir entre serveis. A la Figura 134 es mostra el servei de localitzacions i a las 135 el d'escoles i càmpings.

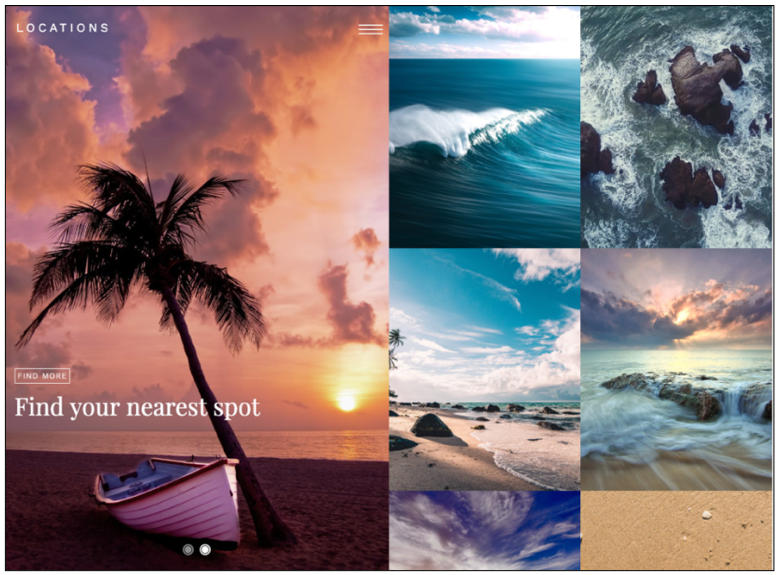


Figura 134 – Servei de Localitzacions

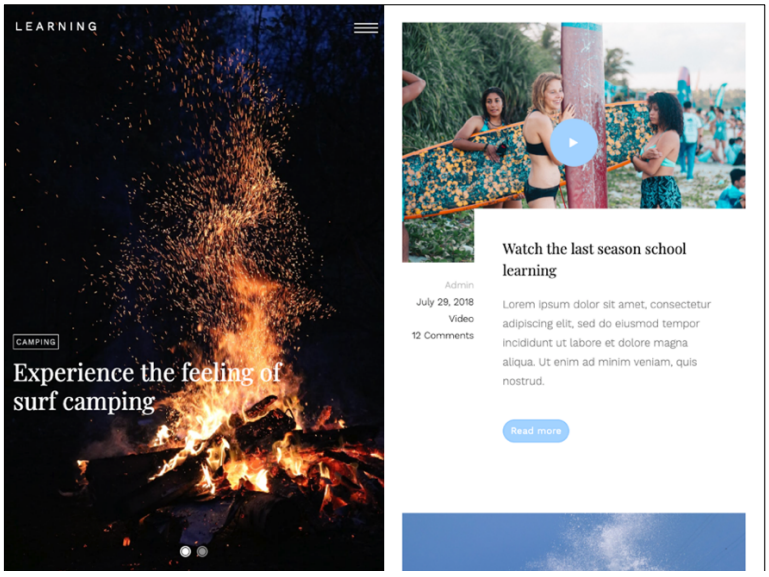


Figura 135 – Servei d'escoles i campings

Les imatges i textos són de moment ficticis i s'han obtingut de la pàgina www.pexels.com i www.es.lupsum.com

5.1.3.8 Responsive Design

Al usar les classes de Bootstrap creant columnes, els elements de la pantalla s'adapten a la mida de la pantalla recol·lectant-se.

En cas de necessitar algun canvi al css es pot editar el fitxer bootsrap.min.less o bé afegir noves regles. Si aquestes reglen només han d'aplicar-se en cas d'accedir des d'un dispositiu de mida de pantalla petita, es pot usar la propietat *media rule.* Per exemple:

```
@media only screen and (max-width: 600px) {
        body {
            background-color: lightblue;
        }
}
```

A l'exemple superior s'aplica una regla CSS només quan la pantalla està per sota de 600px d'amplada.

A continuació es mostren algunes captures dels elements reubicats. Per a simular una mida de pantalla, s'usa l'inspeccionar de Google Chrome i es clica sobre el botó amb una icona de diverses pantalles. Automàticament es mostrarà la pàgina en format mòbil. (Figura 136)

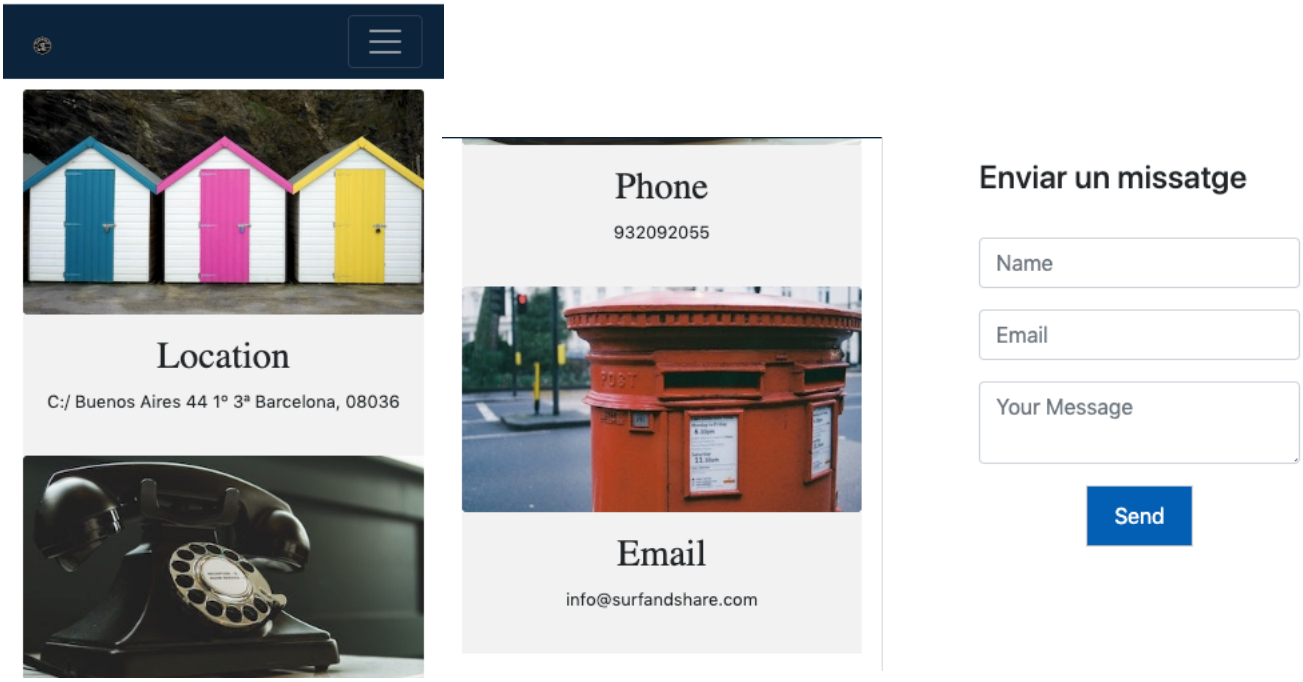


Figura 136 – Alguns elements adaptats a la mida de pantalla

5.1.3.9 E-commerce

El e-commerce és sens dubte una de les parts més rellevants de l'aplicació i la més complicada de dur a terme. Per això, s'ha implementat en un nou entorn virtual i usant noves eines. En concret, s'ha usat Wagtail, un CMS de Python combinat amb Django que incorpora eines i entorns gràfics per a facilitar la creació de llocs com botigues virtuals.

Per a crear un nou entorn virtual cal seguir les següents passes al terminal inferior de Pycharms:

- deactivate. Per a sortir de l'entorn virtual on ens trobàvem
- cd .. Per a tornar enrere al directori arrel del projecte
- virtualenv envShop -p python3. Es crea el nou entorn virtual (envShop)
- source envShop/bin/activate. S'activa el nou entorn virtual
- pip install wagtail. S'instal·la el CSM wagtail
- wagtail start shop. Es crea la nova aplicació (shop)
- cd shop. Entra al directori de la nova aplicació creada
- pip install.r rquirements.txt. Comanda de setup per a preparar el projecte
- ./manage.py migrate. S'apliquen els canvis
- ./manage.py createsuperuser. Es crea un usuari administrador de l'aplicació
- ./manage.py runserver. Es desplega el servidor.

A la Figura 137 es mostra com queda la estructura de fitxers i a la Figura 138 el resultat final de desplegar el servidor i accedir a la url 127.0.0.1:8000 un cop seguits els passos.

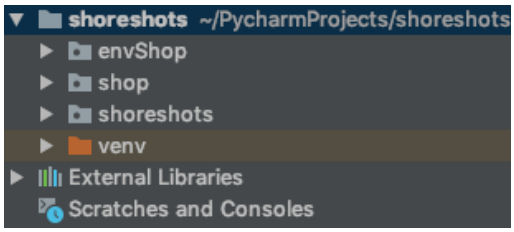


Figura 137 – Nova estructura de fitxers

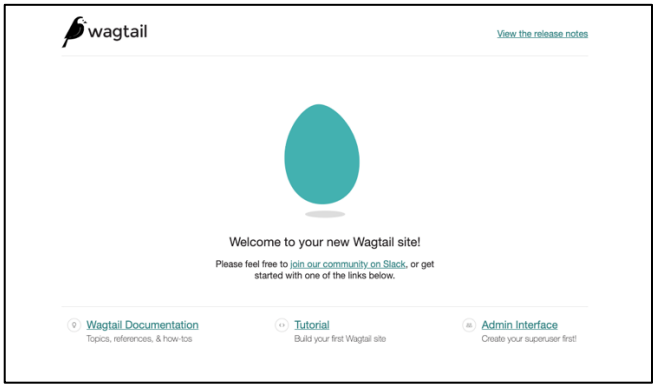


Figura 138 – Wagtail desplegat

Per a activar l'entorn anterior, cal desactivar el actual, i activar l'anterior amb els passos documentats anteriorment.

Integració

Cal mencionar, que tot i estar dins del mateix projecte, les dues aplicacions pertanyen a entorns virtuals diferents. No és el cas descrit inicialment on les aplicacions es registren al fitxer Settings.py i es pot accedir d'una a l'altre mitjançant urls definides.

En aquest document no s'explicarà la integració dels dos sistemes però si es plantegen opcions viables. La primera, migrar el contingut a una aplicació nova dins el mateix entorn, per seguretat no s'ha fet perquè utilitzen mètodes diferents (Django, Wagtail), en principi no ha de passar res però per seguretat s'han separat en dues aplicacions diferents.

La segona opció, en cas de no poder dur a terme la primera o que sigui més complexa de lo esperat, consisteix en desplegar la primera aplicació (polls) al domini principal del servidor contractat i la segona (shop) en un subdomini. De manera que per accedir a la botiga es podria fer via enllaç accedint, per exemple, a www.shop.shoreshots.com

5.2 WebService

Per a crear el Web Service, s'han vist moltes tecnologies disponibles. S'ha optat per a realitzar el desenvolupament amb Django usant el Framework específic Django Rest Framework per als següents motius:

- Ja s'està usant Django per al desenvolupament del projecte.
- És eficaç, simple i ràpid
- Porta totes les eines incorporades de forma nativa

El Web Service a desenvolupar d'inici es força senzill, tindrà poques funcions i no hi haurà unes grans quantitats de dades, per tant s'ha optat per a triar la opció REST ja que Django facilita molt el desenvolupament i per aquest cas és una opció més que adient facilitant molt el desenvolupament i sense dependre de les llibreries externes necessàries per a implementar SOAP (soap.lib)

A continuació es detalla el procés seguit per a la configuració i creació del Web Service:

- S'ha creat un nou projecte anomenat Web Service
- Sobre aquest projecte s'ha creat un nou entorn virtual usant l'assistent de configuració de nous projectes.
- Des del terminal, s'ha activat l'entorn virtual creat: *source venv/bin/activate*
- S'ha instal·lat Django i Django REST: *pip install django, pip install djangorest*
- S'ha creat un projecte: *django-admin.py startproject api* .
- Dins el projecte s'ha creat una app: *django-admin.py startapp shoreshots*
- S'ha configurat el projecte: python manage.py migrate
  python manage.py createsuperuser

Amb aquests passos es crea una estructura de fitxers com la que es mostra a la Figura 139

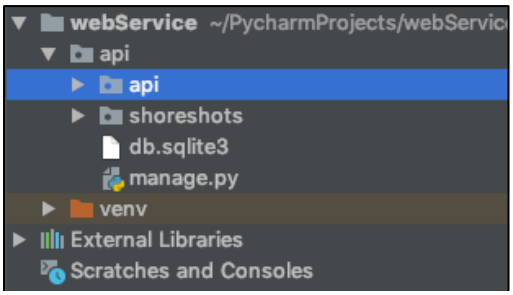


Figura 139 – Estructura de fitxers

Després d'usar el terminal per a la configuració del projecte es pot començar amb el contingut. A continuació es detallen els passos seguit per a la creació del Web Service:

- S'ha afegit l'aplicació creada al fitxer api/Settings.py:

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'shoreshots'
]
```

- S'ha creat el fitxer shoreshots/urls.py

```python
from django.urls import path
from .views import ListUsersView


urlpatterns = [
    path('users/', ListUsersView.as_view(), name="users-all")
]
```

- S'ha definit una url per accedir a l'aplicació api/views.py

```python
from django.contrib import admin
from django.urls import path, re_path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    #url(r'shoreshots/', include('shoreshots.urls'))
    #path('', include('shoreshots.urls'))
    re_path('api/(?P<version>(v1|v2))/', include('shoreshots.urls'))
]
```

- S'ha creat un test per comprovar el funcionament amb una funció que retorna tots els usuaris shoreshots/tests.py

```python
class BaseViewTest(APITestCase):
    client = APIClient()

    @staticmethod
    def create_user(name="", lastName="", username="", email="", password="", type=""):
        if name != "" and lastName != "" and username !="" and email != "" and password != "" and type != "":
            Users.objects.create(name=name, lastName=lastName, username=username, email=email, password=password,

    def setUp(self):
        # add test data
        self.create_user("Sandro", "Del Val", "uskus", "sandro225@hotmail.com", "Buenosaires4413", "surfer")
        self.create_user("Clara", "De Caralt", "pingui", "claradecaralt@gmail.com", "Buenosaires4413", "photo")
        self.create_user("Matias", "Pons", "mathew", "matiesponsmir@gmail.com", "Buenosaires4413", "photo")
class GetAllUsersTest(BaseViewTest):

    def test_get_all_users(self):
        """
        This test ensures that all songs added in the setUp method
        exist when we make a GET request to the songs/ endpoint
        """
        # hit the API endpoint
        response = self.client.get(
            reverse("users-all", kwargs={"version": "v1"})
        )
        # fetch the data from db
        expected = Users.objects.all()
        serialized = UsersSerializer(expected, many=True)
        self.assertEqual(response.data, serialized.data)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

- S'ha creat el model amb la taula Usuaris que emmagatzemarà les dades recollides al formulari de registre

```python
from django.db import models


class Users(models.Model):
    # song title
    name = models.CharField(max_length=255, null=False)
    # name of artist or group/band
    lastName = models.CharField(max_length=255, null=False)
    username = models.CharField(max_length=255, null=False)
    email = models.CharField(max_length=255, null=False)
    password = models.CharField(max_length=255, null=False)
    type = models.CharField(max_length=255, null=False)

    def __str__(self):
        return "{} - {}".format(self.name, self.lastName, self.username, self.email, self.password, self.type)
```

- S'han aplicat els canvis de nou usant el terminal: *python manage.py makemigrations, python manage.py migrate*
- S'ha creat un Serializer els quals converteixen dades complexes com el retorn de la base de dades en models de dades Python que es poden renderitzar en JSON o XML

```python
from rest_framework import serializers
from .models import Users


class UsersSerializer(serializers.ModelSerializer):
    class Meta:
        model = Users
        fields = ("name", "lastName", "username", "email", "password", "type")
```

- S'ha creat la vista que retornarà els usuaris

```python
from rest_framework import generics
from .models import Users
from .serializers import UsersSerializer


class ListUsersView(generics.ListAPIView):
    """
    Provides a get method handler.
    """
    queryset = Users.objects.all()
    serializer_class = UsersSerializer
```

- S'ha executat el test per a comprovar el correcte funcionament: *python manage.py test*

```
Ran 1 test in 0.037s

OK
Destroying test database for alias 'default'...
(venv) MacBook-Pro-de-Marc:api marctulaguardiola$
```

Des del panell d'administrador es poden afegir usuaris manualment tal i com s'observa a la Figura 140

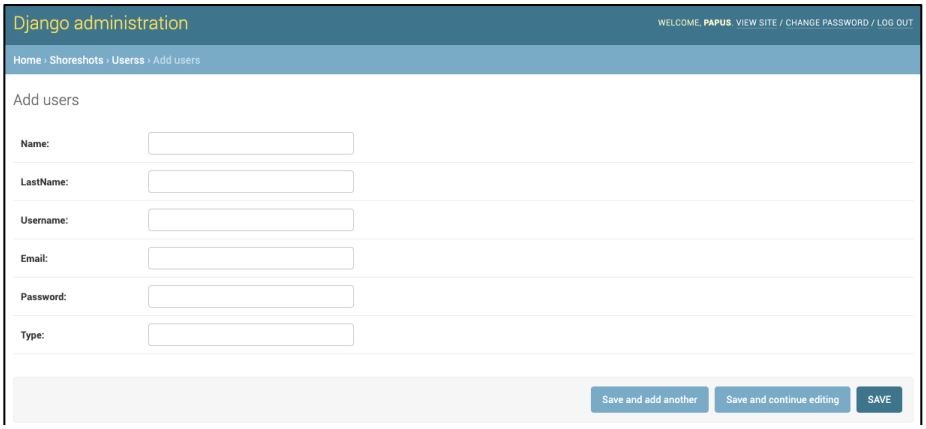


Figura 140 – Panell administrador http://127.0.0.1:8000/admin

Al afegir usuari es pot comprovar el funcionament de la crida de la funció GET amb la url creada http://127.0.0.1:8000/api/v1/users com s'observa a la Figura 141

Figura 141 – Retorn amb la funció GET en format JSON dels usuaris inserits

Realitzats els tests, es hora de realitzar la connexió entre l'aplicació web i el Web Service. Primer de tot cal desplegar els dos servidors, per defecte Django els desplega sobre el port 8000 per tant, s'ha d'indicar un port diferent en uns dels casos.

- Python manage.py runsersver 8002 per a desplegar l'aplicació web a la url: http://127.0.0.1:8002/polls

- Python manage.py runserver per a desplegar el web service a la url: http://127.0.0.1:8000

Des del fitxer views.py de l'aplicació web, en una de les funcions definides (en aquest cas s'ha provat a la funció index) es realitza la connexió tal i com es mostra a la Figura 142. Prèviament caldrà instal·lar el paquet requests: *pip install requests* des del terminal e importar el paquet des del fitxer views.py: *import requests*

```
response = requests.get('http://127.0.0.1:8000/api/v1/users')
data = response.json()
print(data)
```

Figura 142 – Connexió amb el Web Service funció GET

Al mostrar pel terminal les dades obtingudes s'observen els usuaris inserits tal i com es mostra a la Figura 143

```
[{'name': 'Marc', 'lastName': 'Tula', 'username': 'papus99', 'email': 'tulaguardiolamarc@gmail.com', 'password': '7fbc653e835335c318a6eb574f58c163', 'type': 'Surfer'}, {'name
': 'Sandro', 'lastName': 'Del Val', 'username': 'uskus', 'email': 'sanro225@hotmail.com', 'password': '7fbc653e835335c318a6eb574f58c163', 'type': 'Photo'}]
[04/Apr/2019 15:01:11] "GET /polls/ HTTP/1.1" 200 13394
```

Figura 143 – Resposta Web Service format JSON

Per a la recuperació d'imatges s'ha seguit el mateix procediment creant un model i serializer per les imatges i una classe al fitxer views.py així com una nova url. A continuació es detalla la creació d'altres funcions d'inserció com per exemple el registre d'usuaris.

La inserció es realitza per mitjà de la funció POST. S'ha aprofitat la classe anterior creada per als usuaris i s'ha afegit la definició de la funció POST com es veu a la Figura 144.

```
def post(self, request):
    user = request.data.get('user')

    # Create an article from the above data
    serializer = UserAddSerializer(data=user)
    if serializer.is_valid(raise_exception=True):
        user_saved = serializer.save()


    return Response({"success": "User '{}' created successfully".format(user_saved.name)})
```

Figura 144 – Definició de la funció POST

Tal i com s'observa a la Figura anterior**,** es recuperen els paràmetres en format JSON (user) i s'usa un nou Serializer per a la seva encapsulació. A la Figura 145 s'observa el nou Serializer.

```
class UserAddSerializer(serializers.ModelSerializer):
    class Meta:
        model = Users
        fields = ("name", "lastName", "username", "email", "password", "type")
    name = serializers.CharField(max_length=120)
    lastName = serializers.CharField(max_length=120)
    username = serializers.CharField(max_length=120)
    email = serializers.CharField(max_length=120)
    password = serializers.CharField(max_length=120)
    type = serializers.CharField(max_length=120)


    def create(self, validated_data):
        return Users.objects.create(**validated_data)
```

Figura 145 – UserAddSerializer

Per a fer la prova es pot usar la mateixa interfície d'administrador de Django REST Framework com s'observa a la Figura 146.

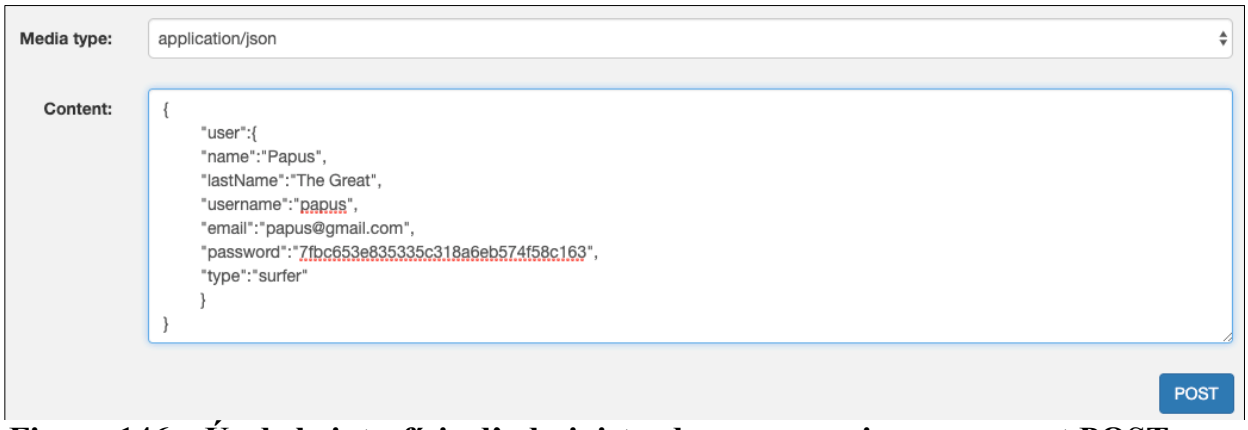


Figura 146 – Ús de la interfície d'administrador per a enviar un request POST

Un cop enviat el request s'observa una resposta afirmativa (200) per part del servidor i el nou usuari inserit. (Figura 147)

Figura 147 – Nou usuari disponible

Cal destacar que per a permetre insercions s'han de modicar els permisos al fitxer Settings.py: *'rest_framework.permissions.AllowAny'*.

Per a fer comprovacions també s'ha usat el servei Postman el qual permet enviar peticions al servidor i observar la resposta. A la Figura 148 es pot observar una petició GET.

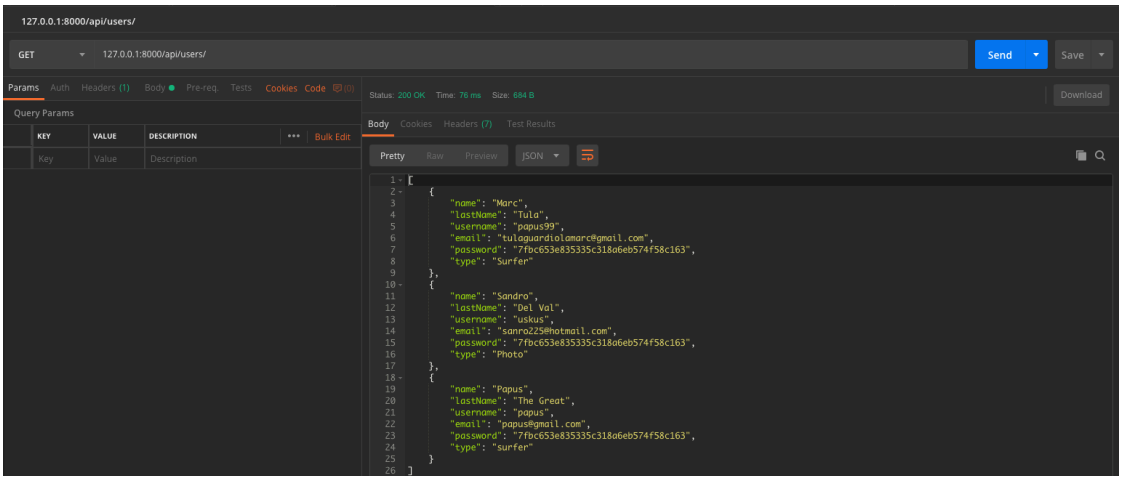


Figura 148 – Postman per al testing del Web Service

5.4 Aplicació Mòbil

5.4.1 Disseny

Per al desenvolupament de l'aplicació s'ha seguit la mateixa guia d'estil que a la pàgina web. És a dir, s'han usat les mateixes games de colors per a mantenir una mateixa imatge corporativa.

En aquest cas no ha sigut necessari l'ús de mockups ja que el propi Android Studio presenta eines visuals per a la visualització de la pantalla en construcció. Un cop definit un disseny bàsic, s'ha traslladat a paper per a fer petits testos d'usuari.

Cal tenir en compte que el target principal és la població practicant d'aquest esport i els fotògrafs involucrats pel que estem parlant d'un sector de entre 18 i 35 anys. En aquest sector, es busca velocitat, és a dir, l'aplicació no ha de tenir un excés d'informació sinó que ha de permetre trobar i realitzar les accions el més ràpid possible.

Actualment existeixen aplicacions mòbil que serveixen per a realitzar els tests d'usuari sense que s'hagi de fer a paper. L'aplicació demana una fotografia del disseny i sobre la imatge es poden identificar els botons per a que quan l'usuari cliqui a sobre simuli l'efecte de l'aplicació al obrir una nova imatge. Un exemple d'aquestes aplicacions és PopApp.

5.4.4 Diagrama de Classes

A la Figura 149 es mostra el diagrama de classes simplificat de l'aplicació a falta d'ampliacions i noves funcionalitats. En ell, es descriuen les diferents classes que intervenen així com la seva estructura i relacions entre elles.

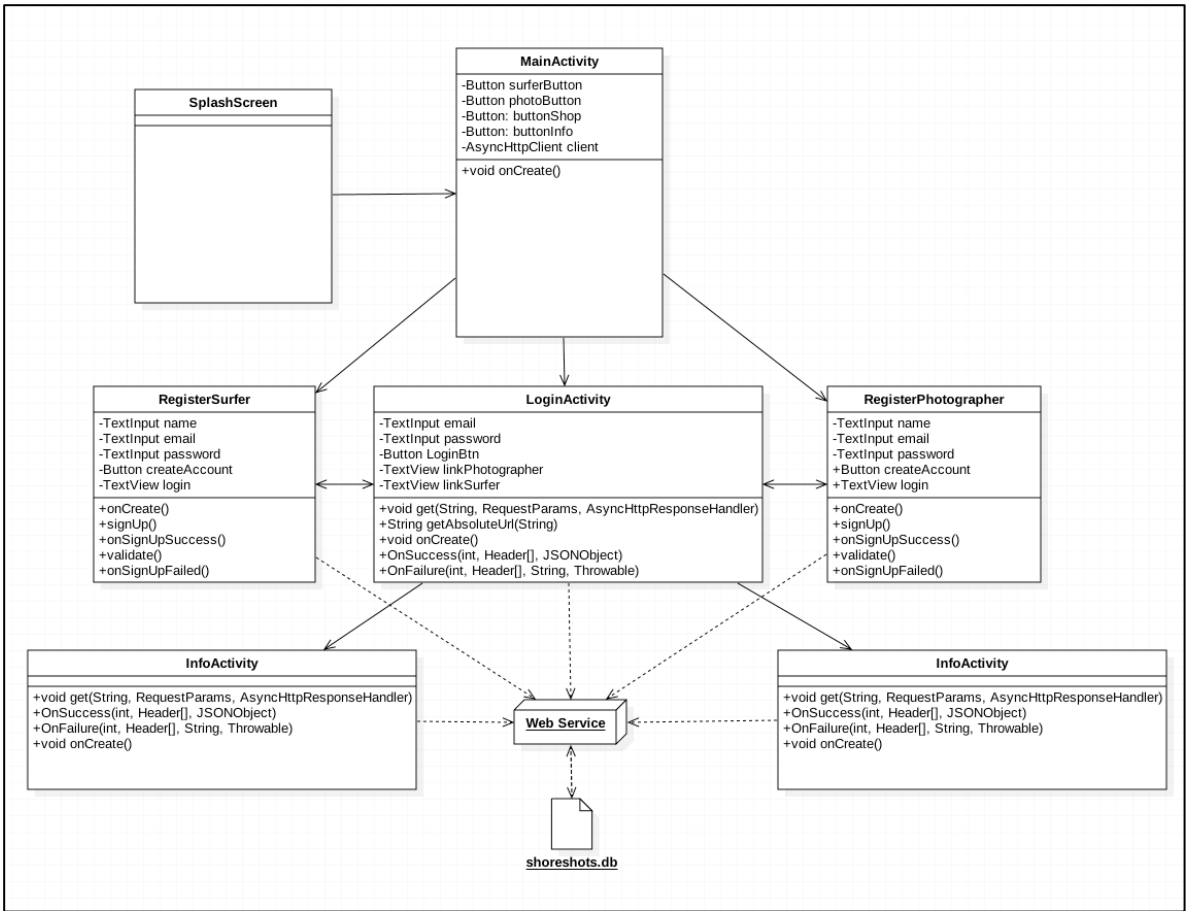


Figura 149 – Diagrama de classes aplicació mòbil

5.4.3 Desenvolupament

En aquest apartat, s'expliquen les diferents pàgines i funcionalitats creades. Cal recalcar que és una aplicació en construcció. Com ja s'ha explicat anteriorment una aplicació Android és un conjunt d'Activities on es codifica el comportament d'una vista associada definida en XML.

5.4.3.1 Manifest

El fitxer Manifest de l'aplicació és el que defineix els elements i configuració que formen la pàgina. Totes les activities s'han de definir en aquest fitxer així com els permisos necessaris (Càmera, internet, etc.), SDK usada, etc.

A la Figura 150 es mostra un fragment del fitxer Manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.marctulaguardiola.shoreshots">
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/logo"
        android:label="@string/app_name"
        android:roundIcon="@drawable/logo"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".SplashActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity" />
        <activity android:name=".RegisterSurfer" />
        <activity android:name=".RegisterPhotographer" />
        <activity android:name=".Info" />
        <activity android:name=".Shop" />
        <activity android:name=".Login"></activity>
    </application>

</manifest>
```

Figura 150 – Android Manifest.XML

5.4.3.2 Splash Screen

Aquesta activitat serà la que es visualitzarà al iniciar l'apliaciació. Per a dur-ho a terme, primer cal definir al Manifest que aquesta serà la Activity principal.

```
<activity android:name=".SplashActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
```

Un cop definida com a activitat principal, s'ha creat el disseny afegint el logo com a imatge central (Figura 151) i programat la activitat per a que salti a la següent pantalla passats 2 segons:

```
new Handler().postDelayed(new Runnable(){
    public void run(){
        Intent intent = new Intent(SplashActivity.this, MainActivity.class);
        startActivity(intent);
        finish();
    };
}, DURACION_SPLASH);
```

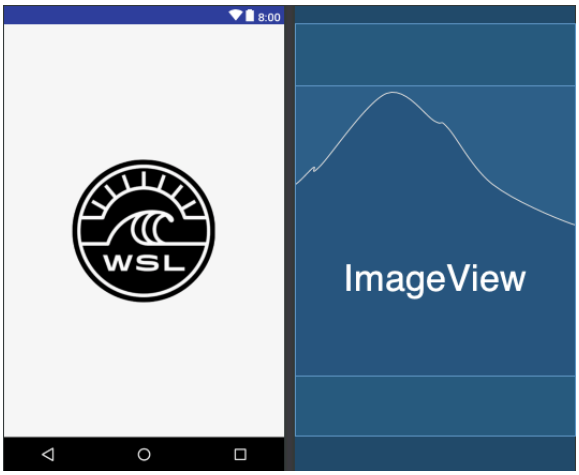


Figura 151 – Splash Activity Android Studio

5.4.3.3 Home

El Splash Screen portarà automàticament al Home passats 2 segons. Al home, es mostra una imatge del surf i 4 botons principals per a accedir ràpidament als serveis (Registre Surfista, Registre Fotògraf, Login al Sistema i Informació). A la Figura 152 es mostra la pantalla principal des del simulador d'Android Studio, en aquest cas un Nexus 5.

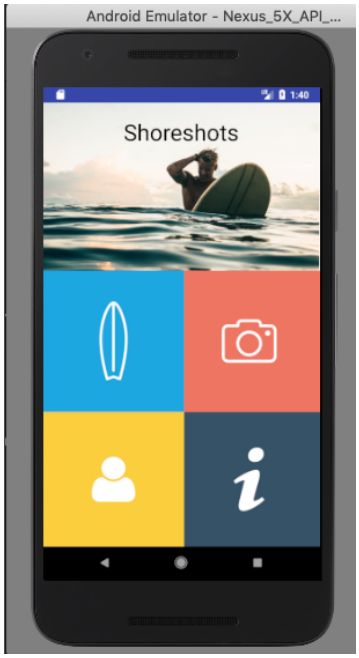


Figura 152 – Main Activity

5.4.3.4 Registre

Es podrà accedir al registre de surfista o al de fotògraf on els camps a introduir variaran lleugerament. Des d'ambdues pàgines es podrà accedir a la pàgina de Login. (Figura 153)

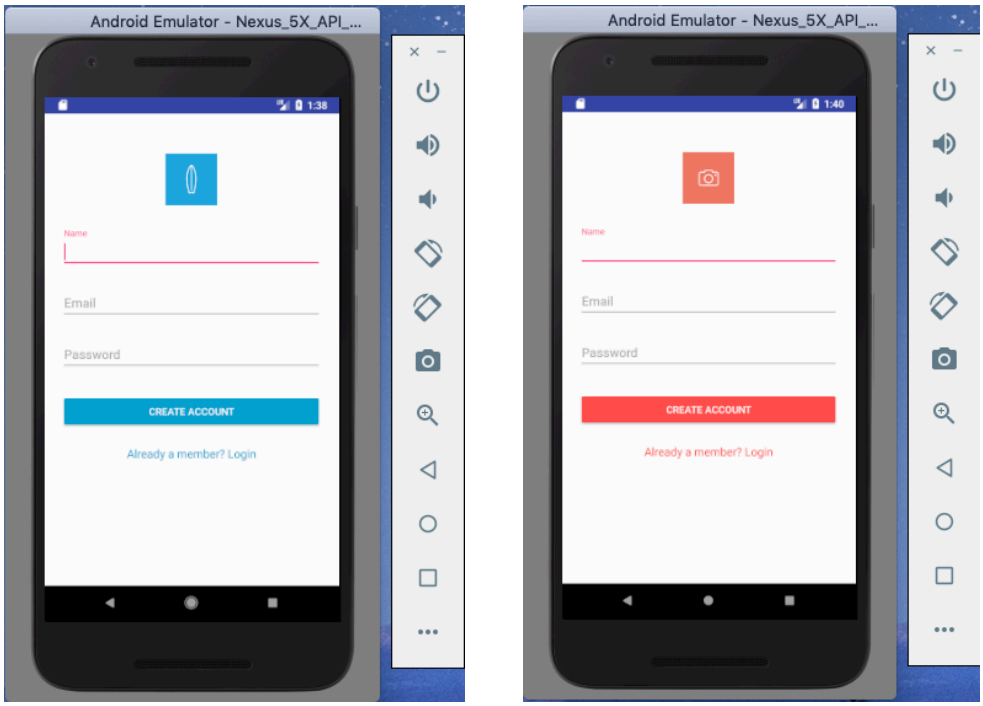


Figura 153 – Pàgines de Registre

Per a la inserció de l'usuari es recuperen les dades i s'envien com a paràmetres a una funció del Web Service.

5.4.3.5 Login

La pàgina de Login recupera tots els usuaris registrats al sistema mitjançant una funció del Web Service i comprova que els paràmetres entrats al formulari coincideixin. També incorpora accessos directes a les pàgines de Registre. A la Figura 154 es mostra l'aspecte de la pàgina de Login.

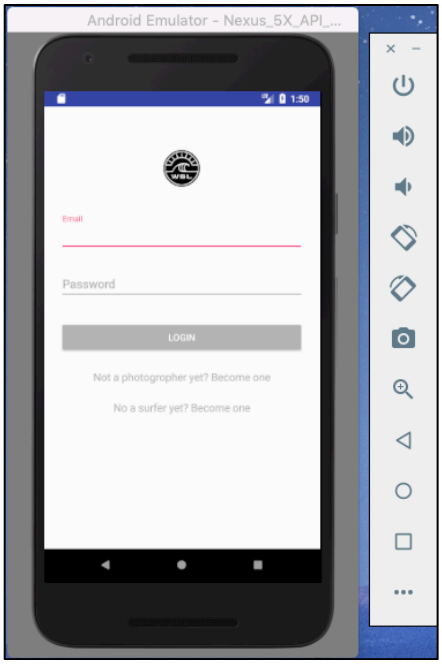


Figura 154 – Pàgina de Login

5.4.3.6 Informació

La pàgina d'informació te la mateixa funció que la de About Us a l'aplicació web. S'explica que és i com funciona Shoreshots així com informació dels usuaris registrats, imatges venudes, etc. (Figura 155)

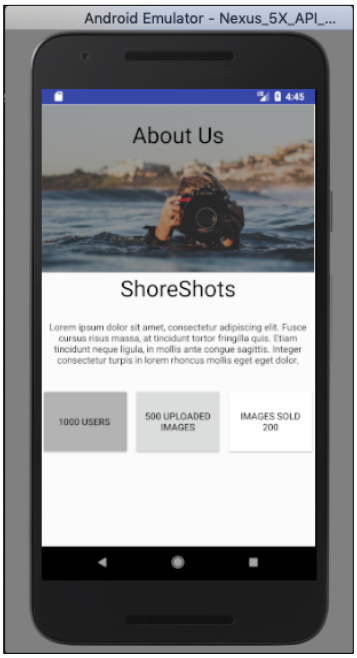


Figura 155 – Pantalla About us

5.4.3.7 Menú després de Login

Un cop comprovat mitjançant Web Service que el Login es correcte s'accedeix a una nova pàgina principal amb el mateix format però mostrant noves funcionalitats (e-commerce, pujar imatges, Serveis i actualització de perfil). A més la pàgina mostra un títol de benvinguda amb el nom d'usuari registrat. Figura 156.

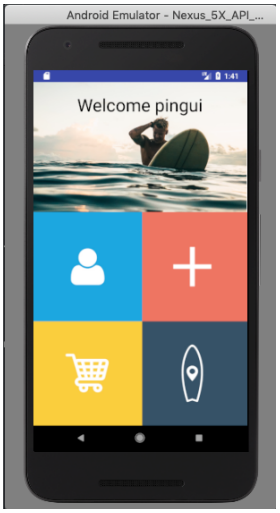


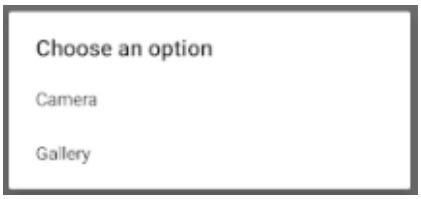
Figura 156 – Menú després de login

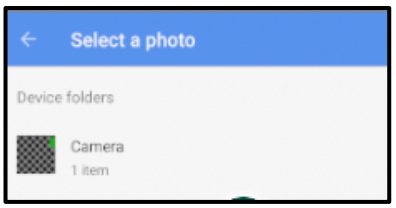
5.4.3.8 Afegir Imatge

Per a pujar una imatge s'ha implementat un accés a la càmera per a poder fer una fotografia o accés a la galeria per agafar una d'antiga. Primer de tot s'han de demanar permisos al manifest per a poder accedir al emmagatzemat i a la càmera.

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Quan l'usuari accedeix a la funcionalitat accepta els permisos i llavors es mostra un *Dialog* per a que esculli entre càmera o galeria. En aquest cas es mostra la galeria perquè el simulador no disposa de càmera. (Figures 157 i 158)





Figures 157 i 158 – Menú d'opcions i accés a la galeria

Un cop seleccionada una imatge aquesta es mostra a l'*Activity* relacionada. Figura 159. Pel que fa a la funcionalitat de serveis, al haver de mostrar tanta informació, de moment s'ha optat per a redireccionar l'aplicació a la web, on l'apartat de serveis està perfectament adaptat a pantalles mòbil (Figura 160)

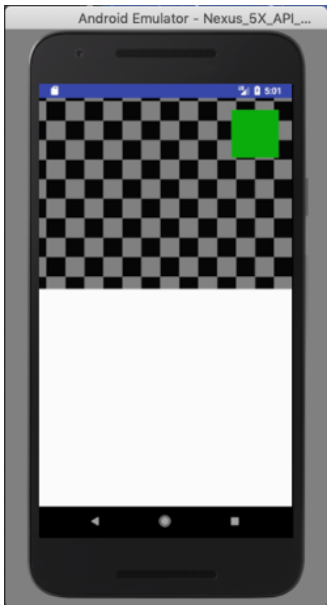


Figura 159 - addImage Activity

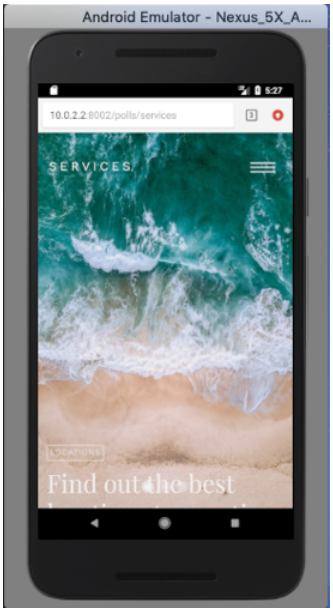


Figura 160 – Pàgina de Serveis

```
Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://10.0.2.2:8002/polls/services"));
startActivity(browserIntent);
```

La pantalla de perfil consisteix en un formulari que recupera els valors per defecte de nom, cognom, nom d'usuari, etc. i permet actualitzar-los via Web Service. La funcionalitat de e-commerce està encara per a desenvolupar.

5.4.4 Connexió amb el Web Service

Per a realitzar la connexió amb el Web Service s'ha usat una llibreria externa anomenada Asynchronous Http Client. Aquesta llibreria permet establir connexió amb un servidor i rebre objectes JSON. Sobreescriu els següents mètodes per a establir un comportament segons el tipus d'objecte i l'èxit o error en la connexió. (Figura 161)

```
AsyncHttpClient client = new AsyncHttpClient();
client.get("https://www.google.com", new AsyncHttpResponseHandler() {

    @Override
    public void onStart() {
        // called before request is started
    }

    @Override
    public void onSuccess(int statusCode, Header[] headers, byte[] response) {
        // called when response HTTP status is "200 OK"
    }

    @Override
    public void onFailure(int statusCode, Header[] headers, byte[] errorResponse,
        // called when response HTTP status is "4XX" (eg. 401, 403, 404)
    }

    @Override
    public void onRetry(int retryNo) {
        // called when request is retried
        }
});
```

Figura 161 – Mètodes per recuperar les dades del Web Service

Cal destacar que des del simulador d'Android Studio no és possible establir connexió amb la localhost 127.0.0.1 sinó que s'haurà de substituir per l'adreça 10.0.2.2. Per tant la URL de connexió serà http://10.0.2.2:8000/api/v1/users.

A la Figura 162 es mostra la codificació per a la obtenció i validació de les dades a la funció *onSuccess* un cop establerta la connexió amb el Web Service.

```
@Override
public void onSuccess(int statusCode, Header[] headers, JSONArray timeline) {
    // Pull out the first event on the public timeline
    Boolean emailOK = false;
    Boolean passOK = false;
    String username = null;
    emailField = (EditText) findViewById(R.id.input_email);
    for (int i = 0; i < timeline.length(); i++) {
        JSONObject user = null;
        String email = null;
        try {
            user = timeline.getJSONObject(i);
            email = user.getString( name: "email");
            username = user.getString( name: "username");
        } catch (JSONException e) {
            e.printStackTrace();
        }
        if (email.equals(emailField.getText().toString())) {
            emailOK = true;
        }
    }
    passwordField = (EditText) findViewById(R.id.input_password);

    for (int i = 0; i < timeline.length(); i++) {
        JSONObject user = null;
        String password = null;
        try {
            user = timeline.getJSONObject(i);
            password = user.getString( name: "password");
        } catch (JSONException e) {
            e.printStackTrace();
        }
        if (password.equals(passwordField.getText().toString())) {
            passOK = true;
        }
    }
```

Figura 162 – Funció OnSuccess per a recuperar les dades.

6 Costos del Projecte

En aquest apartat es descriuen els costos temporals de cada una de les fases més rellevants del projecte. Aquestes fases més rellevants són:

- **Recerca i documentació**: Estudiar les diferents tecnologies, patrons de dissenys, punts forts i febles, etc. Per a implementar un sistema distribuït amb aplicació web i mòbil connectades a un Web Service.

- **Desenvolupament**: Disseny i implementació de l'aplicació web, aplicació mòbil, Web Service i base de dades

- **Redacció de la memòria**: Descriure i especificar el projecte, comparar-lo amb projectes similars, documentar la recerca i la implementació, etc.

Recerca i Documentació

És la etapa que més temps ha comportat. Al seu un projecte tant ampli, hi ha moltíssimes opcions i diverses opinions sobre cada una d'elles. Contrastar i llegir tota la informació ha requerit de moltes hores. Per altra banda, tots els apartats del treball requereixen d'un temps de recerca. L'estat de l'art requereix investigar a la competència i el desenvolupament aprendre les tecnologies que no s'havien usat mai, en aquest cas, Django.

Desenvolupament

Pràcticament ha ocupat tant de temps com la primera, principalment, perquè la majora de tecnologies aplicades no s'havien usat anteriorment i per tant cada pas de la implantació ha requerit més temps de lo normal:

- El disseny de la pàgina web des de els mockups fins als fitxers finals amb css i la implementació Python, html i javascript de totes les vistes i controladors.

- Disseny de la aplicació mòbil des dels tests d'usuari a paper fins a la creació de tots els fitxers i lògica.

- Disseny del web service i implementació de funcions amb les crides a la base de dades, formació de *queris*, encapsulat de retorn, etcètera.

- Disseny de la base de dades des del model conceptual amb totes les entitats i relacions, passant pel model entitat-relació amb les taules i atributs fins al model físic creat amb Django.

En aquest apartat cal tenir en compte que el projecte està en fase de desenvolupament i moltes de les funcionalitats estan encara inacabades. En aquest document s'han explicat les bases de les fases més importants per a la realització del projecte complert.

Redacció de la memòria

Estructurar el document, resumir tota la informació trobada, generar les captures per a mostrar els passos i redactar de forma correcte ha necessitat també de moltes hores.

En total el projecte ha ocupat unes 580 hores tal i com es mostra a la següents Figures 163 i 164.

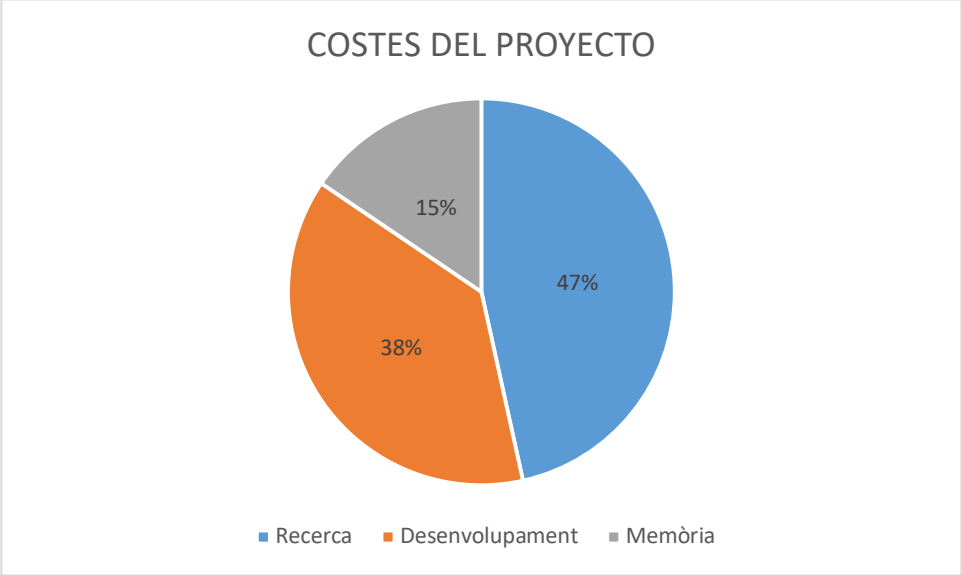


Figura 163 – Costos del Projecte

Fase	Hores	%Total
Recerca	270	46,55%
Implementació	220	37,93%
Memòria	90	15,52%
	580	

Figura 164 – Costos del projecte en hores

7 Línies de Futur

En aquest apartat es proposen diverses opcions per tal d'ampliar el projecte de cara al futur.

Per començar cal dir que a l'apartat de desenvolupament cal ampliar funcionalitats tant a l'aplicació web com al Web Service, aplicació mòbil i base de dades. L'apartat visual es pot millorar així com l'adaptació dels elements a les diferents mides de pantalla.

En el mateix apartat, cal pujar el web service i l'aplicació web a un servidor d'una companyia proveïdora de serveis com pot ser CDmon i l'aplicació a Google Play un cop terminada. Per altre banda seria recomanable el desenvolupament d'una aplicació iOS de cara a cobrir totes les plataformes dels Smartphones.

En quant al producte en sí, cal definir molt bé les funcionalitats que es volen potenciar per a la diferenciació amb la competència. Cal buscar *partners* per al lloguer de material, ofertes per a escoles de surf i millors localitzacions per a la pràctica d'aquest esport.

Cal treballar l'aspecte de SEO incorporant descripcions a les pàgines i imatges, la inclusió de paraules clau als textos i títols, generar el fitxer robots.txt i treballar en el link Building amb els partners trobats.

Una altre bona opció és la creació d'un blog dins el portal web amb noticies noves i recomanacions sobre l'esport per aconseguir més visites així com per a poder incorporar més textos amb paraules clau.

S'han d' incorporar diversos idiomes a les aplicacions per a captar usuaris d'altres nacionalitats. La pràctica d'aquest esport és molt comú a altres països i així es prepara també, una possible expansió a altres països en cas d'èxit.

Instal·lar recursos com Google Analytics per la monitorització de la pàgina i poder veure quins elements necessiten d'una millora al comprovar visites, %rebot, temps de sessió etc.

Cal definir una política de preus i monetització que permeti rendibilitzar i expandir el projecte així com cercar maneres de captar fotògrafs com a usuaris. A nivell de funcionalitats es podria estudiar mostrar un servei de meteorologia i de sessions privades amb els fotògrafs.

8 Conclusions

En aquest apartat es mostren les conclusions extretes un cop acabat el Treball Final de Màster. Sens dubte ha sigut un dels projectes més extensos a realitzar, sinó el que més, on s'han posat en pràctica molts coneixements apresos durant la carrera i màster i s'han aprés molts de nous.

L'objectiu d'aquest Treball Final de Màster era el de dissenyar un sistema distribuït comprés per una aplicació web i una aplicació mòbil connectades a un Web Service per a la consulta i inserció de dades a una base de dades després d'una extensa recerca entre metodologies i tecnologies disponibles per a realitzar projectes similars.

Cal mencionar que no era objectiu principal presentar un projecte totalment acabat però si que es tenia intenció d'intentar-ho. Per desgràcia cal dir que no s'han pogut complir tots els requisits del projecte però si una gran majoria i preparar i documentar les bases per la resta.

Pel que fa a l'objectiu principal, el de documentar i comparar les diferents opcions i metodologies per a dissenyar i implementar projectes d'aquest abast, considero que si s'han assolit els objectius inicials. Es tracta d'un tema summament ampli on cada dia apareixen noves opcions i la comunitat te opinions diverses, pel que ajuntar i contrastar la informació ha resultat una tasca força llarga.

Un altre dels objectius principals personals era el d'aprofitar la ocasió per encarar el desenvolupament de l'aplicació web i Web Service amb una tecnologia mai usada i altament demanada i usada al món laboral. Considero que aquest objectiu s'ha assolit al usar Django per a la implementació. Tot i això, si que es veritat que algunes de les funcionalitats no s'han realitzat totalment. Ha sigut necessària molta recerca d'exemples i informació sobre Django, Django Rest, Python i com implementar algunes de les funcionalitats.

De la mateixa manera que amb la web, s'ha usat Django i Python, mai usats anteriorment per al desenvolupament del Web Service i tot i que s'ha d'estendre amb moltes noves funcionalitats i ampliar la base de dades l'objectiu principal si ha estat assolit.

L'aplicació mòbil s'ha implementat al final de tot quan ja es disposava de menys temps i considero que els objectius no s'han assolit com s'esperava. L'aplicació disposa de poques funcionalitats i necessita de moltes més hores de desenvolupament. Finalment no s'ha pogut realitzar l'aplicació per a iOS que tot i no ser objectiu principal si que es volia mirar d'assolir.

Vistos els objectius individuals del Treball Final de Màster a continuació es mostren algunes conclusions personals.

En primer lloc considero molt rellevant la experiència de portar un projecte real i prou important en solitari. He pogut experimentar la rellevància de realitzar un ERS ja que els clients en molts casos no acaben de saber plasmar amb prou detall la seva idea al projecte o realitzen infinits canvis durant la implementació d'aquest.

Tot i que amb l'ERS també s'han afegit alguns canvis que han enrederit el projecte, ha permès acotar molt l'abast del projecte i blindar els apartats més rellevants. És a dir, que els canvis que s'han realitzat són aspectes més de disseny i no a nivell de funcionalitats.

Un altre aspecte rellevant ha estat el posar en pràctica molts conceptes nous que no s'havien estudiat.

El fet de repassar conceptes per a poder incorporar-los al projecte també m'ha agradat ja que ha permès refrescar alguns mètodes que feia temps no es posaven en pràctica. El treballar amb aquests llenguatges sens dubte facilitarà la feina si en un futur cal tornar-los a usar.

La organització i planificació han tornat a ser determinats en l'èxit del projecte. Es considera que en aspecte es podria haver millorat considerablement, però en general si s'ha fet correctament.

Treballar de forma individual en un projecte d'aquestes característiques també ha suposat un handicap per no poder repartir tasques i haver de planificar millor les etapes.

En resum, amb tots els aspectes bons i dolents que s'han extret d'aquest Treball Final de Màster, sens dubte s'ha obtingut una molt bona experiència que ha suposat una millora tant personal com professional.

9 Bibliografia

Ditrendia (2016, 22 de Juliol) Informe Mobile en España y en el Mundo 2016 [Online]
http://www.amic.media/media/files/file_352_1050.pdf

Coders EYE - 11 Best PHP Frameworks for Modern Web Developers in 2019 [Online]
https://coderseye.com/best-php-frameworks-for-web-developers/

Surfterra, You Surf, They Capture, We Connect
[Online]
https://www.surfterra.com/

You Barrel – Encuentra las mejores fotografías haciendo lo que más te gusta. [Online]
https://www.yourbarrel.net/

ISTR (Ingeniería Software y Timepo Real) Standard IEEE std 830-1198 [Online]
https://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf

Rodrigo Gómez (2015, Novembre) Modelo Vista Controlador [Online]
http://rodrigogr.com/blog/modelo-vista-controlador/

DZone, Best Java Frameworks 2016
[Online]
https://dzone.com/articles/7-best-java-frameworks-for-2016

Medium, JavaScript Trends in 2018
[Online]
https://codeburst.io/javascript-trends-in-2018-3fb0077259

TIOBE, The Software Quality Company
[Online]
https://www.tiobe.com/tiobe-index/

Codeigniter,
[Online]
https://www.codeigniter.com/

Apache,
[Online]
http://www.apache.org/

MAMP,
[Online]
https://www.mamp.info/en/

php.net,
[Online]
http://php.net/manual/es/index.php

Javascript,
[Online]
https://www.javascript.com/

W3Schools, AJAX Introduction
[Online]
https://www.w3schools.com/xml/ajax_intro.asp
https://api.jquery.com/jQuery.ajax/

W3Schools, HTML5 Tutorial
[Online]
https://www.w3schools.com/html/

W3Schools, CSS Tutorial
[Online]
https://www.w3schools.com/Css/

W3Schooles, Bootstrap Tutorial
[Online]
https://www.w3schools.com/bootstrap/
Documentació Oficial: http://getbootstrap.com/

Margaret Rouse – TechTarget (2011, Decembre)
[Online]
http://searchmobilecomputing.techtarget.com/definition/iOS

Apple Inc,
[Online]
https://www.apple.com/

Apple Developer, Xcode
[Online]
https://developer.apple.com/xcode/

EcuRed (2003, Maig) Objective – C
[Online]
https://www.ecured.cu/Objective-C

Exemples de implementacions:
[Online]
https://www.tutorialspoint.com/objective_c/objective_c_classes_objects.htm

Swift,
[Online]
https://swift.org/

Júlio Cesar Fernandez, (2017, Agost) Swift y Objective-C
[Online]
https://applecoding.com/analisis/swift-objectivec-analizamos-cual-es-mejor#

Simon Allardice, (2012, Agost) What is Cocoa?
[Online]
https://www.lynda.com/Cocoa-tutorials/What-Cocoa/85872/110013-4.html

Deviservi (2014, Octubre)
[Online]
https://github.com/deviservi/nusoap

mySQLWorkbench,
[Online]
https://www.mysql.com/products/workbench/

phpMyAdmin,
[Online]
https://www.phpmyadmin.net/

LanceTalent, Mejores Herramientas Para Hacer El Prototipo De Tu App
[Online]
https://www.lancetalent.com/blog/mejores-herramientas-prototipo-app/

ValueCoders, Is Ruby on Rails Dead In 2018?,
[Online]
https://www.valuecoders.com/blog/technology-and-apps/ruby-rails-dead-2018/

Chema Alonso (2007 Juliol), Un informático en el lado del mal, SQL Injection
[Online]
http://www.elladodelmal.com/2007/07/sql-injection-level-1.html

OKHosting, Metodologías del Desarrollo de Software
[Online]
https://okhosting.com/blog/metodologias-del-desarrollo-de-software/

Priore (2017, Agost)
[Online]
https://github.com/priore/SOAPEngine

Apple Developer (2015, Agost)
[Online]
https://developer.apple.com/library/content/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CoreServicesLayer/CoreServicesLayer.html

Android Asynchronous Http Client, James Smith 2017
[Online]
http://loopj.com/android-async-http/

Daffodil – Software Development Insights, Top 10 Python Frameworks for Web Development
[Online]
https://insights.daffodilsw.com/blog/top-10-python-frameworks-for-web-application-development

HotFrameworks, Find your new favourite Framework
[Online]
https://hotframeworks.com/

Crunchbase – Here Are The Most Popular Web Frameworks
[Online]
https://news.crunchbase.com/news/popular-web-frameworks-seed-early-stage-startups/

AlexSoft, 28 Jun 2018 – Swift vs Objective C
[Online]
https://www.altexsoft.com/blog/engineering/swift-vs-objective-c-out-with-the-old-in-with-the-new/

[IEEE830] IEEE Std 830-1998 Recommended Practice for Software Requirements

10 Annex

ERS – ShoreShots

Software Requirements Specification

Versión 1

13/10/2018

Marc Tula

Índex ERS

1 Introducción

1.1 Objetivo

Este documento tiene como propósito especificar los requerimientos funcionales y no funcionales del sistema a desarrollar así como su alcance. Se escribe basándose en les directrices del estándar "IEEE Std 830-1998 Requirements and Specifications" [IEEE830].

En el documento se plantean las funcionalidades, diseño, apartados, lógica, etc. Necesarias para el desarrollo del proyecto

El documento va dirigido al futuro equipo de desarrollo.

1.2 Alcance

Se quiere desarrollar un portal de venta de imágenes y otros productos orientado a la disciplina deportiva del surf y otras similares. Fotógrafos registrados en la plataforma captarán con sus cámaras instantáneas de surferos, los cuales se podrán encontrar en la página mediante localización o otros filtros y adquirir la imagen por un precio. A continuación se define la estructura jerárquica a implementar. Más adelante se especificarán las funcionalidades del sistema.

- **Web Service**: Definición e implementación de las funciones que actuarán sobre la base de datos y el sistema (Inserción, borrado, edición i recuperación de los datos)

- **Página Web**: Desarrollo de una página web que permita el registro de usuarios, la creación de perfiles y subida de contenido multimedia. Las funcionalidades más específicas de la página quedarán definidas más adelante.

- **Aplicación Móvil**: Desarrollo de una aplicación móvil para las plataformas de iOS y Android con las mismas funcionalidades que la página web para que los usuarios puedan acceder al sistema de forma más cómoda con los móviles. La aplicación se comunicará con el *Web Service* para ejecutar las funcionalidades definidas.

1.5 Visión general del producto

El producto a desarrollar consiste en un portal de venta de imágenes deportivas relacionadas con el surf y otras disciplinas parecidas. Se ha observado que existen un mercado emergente en este sector explotado por otras plataformas como Surfterra. ShoreShots pretende competir en este mercado complementando las carencias que puedan tener otras plataformas.

Se incentivará a fotógrafos para que acudan a las localizaciones más 'surfistas' para fotografiar a los surferos. Estos fotógrafos se crearán un perfil en Shoreshots donde subirán las instantáneas realizadas con unas etiquetas de descripción (Localización, fecha, etc..).

Los surferos, conocedores del servicio, se buscarán en la plataforma y podrán adquirir las imágenes por un precio después de registrarse. Del importe recibido, una parte será para el fotógrafo y otra para Shoreshots, la cuál pretende también convertirse en una marca referente en el sector deportivo del surf y deportes parecidos ofreciendo otros servicios como una tienda de productos personalizados, noticias relacionadas con estos deportes, guía de los mejores sitios para practicarlos, sitios donde aprender y alquilar material.

El éxito de Surfterra se centra de momento en Portugal. España es otro país con una gran cultura en estas disciplinas deportivas. Shoreshots se centrará, en un principio, en explotar el servicio en territorio español siempre con un modelo estratégico de desarrollo que permita, de una forma sencilla, replicar el servicio en otros países.

2. Descripción General

2.1 Perspectiva del producto

El producto será independiente y totalmente autónomo. La arquitectura del sistema responderá al modelo Cliente-Servidor. Los usuarios accederán al sistema con un ordenador a través del portal web o con el móvil a través de la aplicación.

Existirá un perfil de administrador de la plataforma el cual tendrá permisos para gestionar los usuarios registrados, podrá validarlos o borrarlos. Un usuario se podrá registrar como:

- **Surfero**: Deberá rellenar unos campos básicos de información y validar su perfil mediante un enlace en el correo. En ese momento podrá realizar compras tanto de imágenes como de productos en la tienda.

- **Fotógrafo:** Deberá rellenar un formulario más extenso, con información bancaria y aceptando las políticas de Shoreshots para establecer una relación legal mediante la cual se expresa los porcentajes a recibir por la venta de imagen y las reglas a seguir (Tamaño imagen, precios máximos y mínimos, etiquetas, etc.). De nuevo, deberá validar su perfil mediante un enlace al correo proporcionado y en ese momento podrá subir imágenes. Los fotógrafos podrás crear un perfil con sus imágenes y proporcionar otros servicios suyos (Sesiones privadas)

- **Administrador:** El administrador podrá recolectar datos para realizar estudios de tendencias, etc., siempre con el consentimiento de los usuarios mediante la política de privacidad. Tendrá permisos para gestionar los usuarios registrados, podrá validarlos o borrarlos siempre informando mediante correo electrónico.

La base de datos estará pensada de forma escalable, preparada para abarcar más etiquetas, localizaciones y otros datos significativos.

La conexión cliente - servidor se hará mediante un Web Service y todos los sistemas (Web, Móvil) se conectarán a la misma base de datos.

2.2 Funcionalidades del sistema

2.2.1 Registro y perfil como fotógrafo

Un fotógrafo tendrá que registrarse para crear su perfil y ser accesible mediante consultas sobre la base de datos.

El registro tendrá un formulario con información específica:

- Nombre, Apellidos, Correo, Contraseña, Teléfono, Localizaciones habituales, cuenta bancaria.

El registro se completará una vez el usuario haya validado su perfil mediante el clic en un enlace que recibirá por correo. Dicho enlace pondrá a '1' un campo de validación de la base de datos.

Una vez registrado tendrá acceso vía login a su perfil donde tendrá un espacio grande para subir imágenes. Seguidamente encontrará un campo de texto con sus puntos fuertes y experiencia a nivel de currículum así como su información de contacto.

En este espacio podrá incluir también referencias a trabajos externos para promocionarlos. El usuario deberá entender en todo momento que su perfil será visible en la plataforma por todos los visitantes. No obstante, el usuario tendrá opciones de privacidad para decidir que información muestra.

El usuario también tendrá un apartado para actualizar su perfil y configuración.

2.2.2 Usuarios anónimos

Los usuarios que entren en Shoreshots sin autenticar podrán realizar las siguientes acciones:

- Leer noticias relacionadas con el surf y parecidos.
- Información sobre los mejores campamentos y escuelas para aprender la práctica de estos deportes.
- Mejores sitios para alquiler de material. Recomendación de marcas y equipo necesario.
- Las mejores localizaciones para la práctica de estos deportes y eventos grandes relacionados.
- Ver y buscar imágenes mediante filtros.

- Ver y buscar perfiles de fotógrafos así como sus imágenes. (En un futuro podría restringirse a usuarios registrados)

- Ver los productos de la tienda.

2.2.3 Surferos

Cuando un usuario se registre pasará a ser un surfero. Los surferos podrán realizar las siguientes acciones extras:

- Comprar imágenes.
- Comprar productos y añadir comentarios o reviews.
- Preguntas y dudas relacionadas con el deporte, campamentos, equipo, etc.. El administrador contestará las preguntas pero otros surferos también podrán (foro)
- Si no indican lo contrario, los surferos recibirán campañas publicitarias de Shoreshots, con ofertas de campamentos y escuelas, nuevos productos en la tienda, ofertas de alquiler de material, etc.

2.2.4 Gestión de usuarios

El administrador recibirá una notificación cuando un usuario nuevo se haya registrado y confirmado mediante el enlace al correo electrónico, momento en el que, en caso de ser un fotógrafo deberá acceder al perfil para validar la información. Validar se entiende como comprobar que la información (visual y texto) cumple con lo que se debería mostrar en la plataforma según las condiciones y políticas de Shoreshots. NO es Trabajo de Shoreshots el comprobar que la información relacionada con trabajos externos sea cierta.

En caso de el administrador manifieste que la información proporcionada por el usuario no cumple con las políticas y condiciones de Shoreshots podrá eliminar el perfil, proceso que se notificará al usuario afectado vía correo electrónico explicando los motivos (Sistema automatizado).

2.2.5 Mostrar imágenes

Antes de que el usuario visitante, surfero o fotógrafo realice una búsqueda por filtros para encontrar imágenes, se mostrarán unas de muestra. Existen varias opciones para mostrar imágenes de muestra. La más sencilla es mostrar una selección (10, 15, 20) ordenadas por fecha, es decir, se mostrarían las últimas.

Otra opción, más complicada, va relacionada con la valoración de las imágenes por parte de los surferos. Si se decide que los surferos podrán puntuar las imágenes (Estrellas, nota 1-5, etc..) se podría hacer una selección para mostrar de las imágenes

mejor valoradas. Esta opción también podría generar una competencia positiva entre fotógrafos

2.2.6 Compra de imágenes o productos

Los surferos podrán comprar imágenes y productos de la tienda (Fotógrafos también). Se necesitará un certificado SSL para garantizar la correcta encriptación de los datos introducidos (Tarjetas, etc). La plataforma de pago será REDSYS con lo que se deberá hablar con el banco (Sabadell) para la obtención de claves necesarias para la configuración.

En el caso de la compra de imágenes de forma automatizada se generarán dos transacciones. Una primera para Shoreshots la cual corresponde a un porcentaje sobre el precio de la imagen. La segunda transacción será a la cuenta bancaria asociada al perfil del fotógrafo vinculado a la imagen. La cantidad será el montante restante una vez realizada la primera transacción.

2.2.7 Precios Imágenes

Los precios de las imágenes los decidirán los fotógrafos. Eso sí, los valores máximos y mínimos estarán acotados según las políticas de Shoreshots.

2.2.8 Idiomas (A concretar)

El producto original se lanzará en castellano y inglés. Los idiomas disponibles tanto en el portal web como en la aplicación se irán ampliando a medida que existan masas de usuarios representativas con idiomas distintos.

2.2.9 Gestión de contenido multimedia

A medida que el sistema evolucione se irán mejorando los Servicios de hosting para disponer de más ancho de banda y más espacio de memoria. Sin embargo, al principio será necesario definir políticas para la subida de contenido por parte de los fotógrafos. La cantidad máxima de imágenes será limitada. El fotógrafo podrá decidir si borrar una imagen para subir una nueva.

2.2.10 Marcas de agua

Las imágenes deberán incorporar una marca de agua para evitar descargas gratuitas por parte de los usuarios. Estas marcas de agua se generarán de forma transparente al fotógrafo, es decir, cuando suba una, de forma automatizada se generará la marca de agua.

La imagen original será guardada en un directorio privado del servidor. Al efectuarse una compra, se generará un enlace de descarga privado (Más complicado) o se le enviará por correo al surfero (Más simple).

Puede que se muestren miniaturas de las imágenes en visores para mejorar el diseño y la usabilidad. En este caso, las miniaturas tendrán una resolución muy baja para evitar descargas.

2.3 Características del usuario final del portal

El sistema está dirigido a jóvenes deportistas (18 - 40 años). Será importante que la interfaz gráfica diseñada y el funcionamiento general de la página/aplicación sean lo más intuitivas posible. Es por eso que a la página principal dispondrá de un vídeo tutorial donde de mostrará el funcionamiento del sistema así como los pasos para realizar cada una de les acciones.

2.4 Restricciones/Limitaciones

Únicamente podrán crear un perfil y subir contenido los fotógrafos registrados.

La única persona que podrá borrar y modificar usuarios será el administrador.

Los usuarios deberán validar su cuenta mediante un enlace enviado al correo para poder acceder a todo el sistema.

La página web deberá seguir un “Responsive Design” para la correcta adaptación a todos los dispositivos móviles y tablets. Por otra parte no será una prioridad máxima ya que el objetivo es que en móviles y tablets se acceda al sistema mediante una aplicación (Android, iOS).

2.5 Monetización

Shoreshots cobrará un porcentaje de cada transacción que se realice relacionada con la compra de imágenes. Por otra parte mediante se generarán ingresos mediante la venta de productos en tienda.

Casos a estudiar:

- Porcentaje cuando algún usuario de shoreshots reserve en algún campamento o escuela desde la web o con un código.

- Porcentaje cuando algún usuario de shoreshots alquile equipo deportivo desde la web o con un código.

3. Requerimientos Específicos

3.1 Interfaces de usuario

El sistema constará de una página web y una aplicación para las plataformas de iOS y Android como interfaces de usuario. También tendrá unes interfaces web específicas para el administrador.

2. Interfaces software

El sistema interactuará con:

- Un DBMS (Data Base Management System) relacional donde se almacenará la información persistente del sistema.
- Un Web Service donde se encuentran definidas todas las funciones necesarias para la inserción y consulta de datos

Tecnologías: Se valorarán los lenguajes *PHP* y *.Net* para el desarrollo del *Web Service* y el *backend* de la web. El Web Service se implementará en *SOAP*. Para las aplicaciones se usará *Objective C* como lenguaje de iOS y Android nativo para la aplicación Android. El *frontend* de la página web será una combinación entre *HTML, Javascript* y *CSS*. Se estudiarán distintos *frameworks* como *Codeigniter* para el desarrollo del portal web siguiendo la metodología *modelo-vista-controlador*. Se valorará la opción de usar wordpress con una plantilla desde 0. Es decir, se crearía todo el código pero se podría usar plugins específicos.

Necesidades: Es necesario el disponer de un servidor que permita la tecnología *MySQL* y *PHP*. Será necesario dominio y hosting el cual se irá ampliando a medida que aumente el tráfico del sistema. El Web Service y el portal web serán alojados en el mismo servidor junto a la base de datos. También será necesario un certificado SSL para la seguridad del sistema así como cuentas en *Apple Store* y *Google Play* para el desarrollo de las aplicaciones. Se deberá configurar un correo de dominio para el contacto con los usuarios finales del sistema.

Base de datos: La base de datos se construirá de forma escalable con MySQL accesible mediante *phpMyAdmin*. Se deberán crear rutinas de *Backup, RollBack,* migración y *merge* para la seguridad y persistencia de los datos.

Lanzamiento: Para controlar el flujo de tráfico y transacciones en el sistema es recomendable lanzar la aplicación de forma incremental, empezando por ejemplo, en territorio nacional e ir expandiendo.

Sin embargo, se estudiaran tecnologías como *Amazon Web Services* que permitirían replicar el sistema en distintos puntos del mundo para acercar los datos a los usuarios finales mejorando la escalabilidad del sistema, proporcionando réplica de los datos y permitiendo lanzar el producto de forma mundial con un control de flujo estable.

3.3 Interfaces hardware

La aplicación móvil funcionará con todos los dispositivos iOS que soporten las versiones 8.0 y posteriores. Por parte de Android funcionará en todos los dispositivos con SKD 24 o superior.

3.4 Requerimientos no funcionales

3.4.1 Usabilidad

Esta deberá de ser de uso intuitivo y aspecto serio, procurando darle una imagen corporativa. Las aplicaciones seguirán las mismas guías de estilo que la página web y en el caso de iOS cumplirá con todos los estándares de Apple.

La página web deberá de ser intuitiva y clara con una apariencia agradable y adaptable a todos los dispositivos y medidas de pantalla. Los colores deben de formar un diseño harmónico y las fuentes y tamaño del texto deben de permitir una lectura fluida.

La información debe estar dispuesta de tal manera que el usuario pueda encontrarla de forma fácil y rápida, minimizando el número de clics necesarios. Los formularios deben de sintetizar la información el máximo posible para que el usuario se registre cómodamente.

3.4.2 Fiabilidad

En el sistema existirán operaciones críticas como pagos bancarios por lo que el resultado final debe ser seguro, robusto y determinista donde todas y cada una de les funcionalidades han de funcionar correctamente, los posibles errores han de estar correctamente controlados y en ningún caso se aceptará que la aplicación entera o parte de ella acabe como consecuencia de un error. El sistema pasará por una fase exhaustiva de testeo del sistema usando técnicas habituales como son las baterías de test, pruebas unitarias, etc.

3.5 Requerimientos de rendimiento

3.5.1 Tiempo de respuesta

Los tiempos de respuesta de las consultas a la base de datos procedentes de una interfaz de usuario han de ser reducidos, nunca superando el segundo por ninguna operación. Por lo tanto, habrá que dedicarle tiempo a realizar un diseño óptimo de la base de datos así como a *queris* optimizadas.

3.6 Requerimientos de desarrollo

3.6.1 Ciclo de vida

Ciclo de vida en **espiral** ya que se irán aumentando las funcionalidades poco a poco. Esta metodología viene a decir que, en cada iteración de la espiral se definirán los aspectos a implementar, se implementarán, se testearán y se analizará su integración con las iteraciones anteriores. La nueva iteración repetirá los mismos pasos. De esta manera se minimizan los errores de desarrollo y se permite añadir cambios de forma más eficiente.

3.6.2 Plazos

El sistema deberá estar funcionando a finales de 2019.

3.6.3 Control de versiones

Los distintos desarrolladores del proyecto (A definir) trabajarán con un sistema de versiones privado (Ej. Github).